Genetics-based Machine Learning

Tim Kovacs

Dept. of Computer Science
University of Bristol

April 2009

These slides accompany the chapter *Genetics-based Machine Learning* by Tim Kovacs, in Grzegorz Rozenberg, Thomas Bäck, and Joost Kok, editors, a Handbook of Natural Computing: Theory, Experiments, and Applications. Springer Verlag, 2010. Please note each has some references the other does not so the numbers used to cite papers do not match.

Contents

Contents

Outline

This survey:

1. Introduces the subject
    - introduces Supervised Learning (SL)
    - contrasts SL with optimisation
    - assumes readers are familiar with Evolutionary Algorithms (EAs)
    - discusses pros and cons of GBML
2. Describes a framework for GBML
    - classifies forms of GBML (learning, meta-learning etc.)
    - reviews interaction of learning and evolution
    - outlines high-level algorithms
3. Reviews the major forms of GBML
    - with emphasis on evolutionary aspects
    - organised by research community (and not e.g. by learning paradigm)
4. Concludes

What's missing

Coverage is somewhat arbitrary and missing:

- A general introduction to Machine Learning including:
  - Structure of learning problems and fitness landscapes
  - Non-evolutionary algorithms
  - Theoretical limitations (e.g. *no free lunch* theorem for learning)
- Evolutionary methods for:
  - Clustering
  - Reinforcement Learning
  - Bayesian Networks
  - Artificial Immune Systems
  - Artificial Life
  - Application areas

There's also little on:

- EAs for data preprocessing e.g. feature selection
- Comparisons between GBML and non-evolutionary alternatives
- Co-evolution

Machine Learning

ML is about machines which:

- improve with experience
- reason inductively or abductively

In order to:

- optimise
- approximate
- summarise
- generalise from specific examples to general rules
- classify
- make predictions
- find associations
- propose explanations
- propose ways of grouping things
- ...

GBML

- We consider any stochastic search based method as GBML
- Most are population-based
- Most popular are:
  - Genetic Algorithms (GAs)
  - Genetic Programming (GP)

Inductive generalisation

Inductive generalisation:

- Inferring unknown values from known values
- We *assume* they're correlated!
- Objective: to maximise a function of unknown cases
  - Called the fitness function
- There's *no need for induction* if:
  - all values are known, and ...
  - there's enough time to process them
- We consider two forms of induction:
  - function optimisation
  - learning
- We won't deal with abduction

1-max: a typical *optimisation* problem

1-max problem

- Maximise the number of 1s in a binary string of length n
- Optimal solution is trivial for humans

Representation:

- Input: none
- Output: bit strings of length n

Data generation:

- Data: generate as many output strings as you like
  - Time is the limiting factor
  - If time allows you can enumerate the search space O

Training:

- Fitness: number of 1s in output string

Evaluation with 1-max

Evaluation:

- How close did learner get to the *known* optimal solution?
  - 1-max is a toy problem
  - In realistic problems optimum is often not known
  - And we may or may not know maximum possible fitness
- Alternative measures for both toy and realistic problems
  - How much training was needed?
  - How did it compare to other solutions?

Classification: a typical *learning* problem

Classifying mushrooms:

- Given features of each species (colour, size . . .) including whether it is edible
- Learn a hypothesis which will classify new species

Representation:

- Input: a set of nominal attributes for each species
- Output: binary label: 'Poisonous' or 'Edible' for each species

Classification continued

Data generation:

- A fixed data set of input/output examples obtained from an expert on mushrooms
    - $D = \left[(i_1, o_1), \ldots (i_n, o_n)\right]$
- where
    - n is the number of examples
    - n is much smaller than the input space
- Partition D into train and test sets to evaluate generalisation

Training:

- Maximise classification accuracy on train set

Evaluation:

- Accuracy on test set – an indication of how well a new species might be classified

Supervised Learning

- We focus on the primary learning paradigm: standard SL
  - Many others exist
- We have a set of input/output pairs
  - Defining feature of SL: outputs are part of data set
    - Mushroom example was SL
  - Inputs are factored into attributes
  - Divide available data into training and test sets
- Problem is to predict correct output on future data
  - Find correlations between attributes and output on training set
  - We evaluate inductive generalisation on test set
  - Performance on test set assumed indicative of performance on future data

Learning and optimisation compared

Learning:

- Typically have limited training data
- Crucial to get inductive bias right for later use on new data
- Hence *must* evaluate generalisation to unseen cases of *same* problem

Optimisation:

- Typically can generate as much data as time allows
  - Typically any data point can be evaluated
- Hence test set not needed
  - Concerned with finding optimum data point in minimum time
  - Specifically: inducing which data point to evaluate next

Issues in Supervised Learning

- Hypothesis complexity: overfitting, underfitting
- Noise, missing attributes
- Class imbalances (e.g. many poisonous, few edible)
- Learning from one class only
- Biased cost functions (e.g. false positives vs. false negatives)
- Human readability
- Non-stationary functions, online learning, stream mining
- Learning from little data
- Learning when there are too many attributes: feature selection
- Incorporating bias and prior knowledge
- Handling structured data
- Using unlabelled data
- ...

Reasons to use GBML 1

- Accuracy is competitive with other methods ([99] §12.1.1)
- Exploit the synergy of learning and evolution
  - Combine global and local search
  - Baldwin effect smooths fitness landscape
- Combine feature selection and learning
  - E.g. feature selection is intrinsic in LCS
- Adapt inductive bias
  - Representational bias by e.g. selecting condition shapes
  - Algorithmic bias by e.g. evolving learning rules
- Exploit diversity in population
  - to combine and improve predictions (ensemble approach)
  - to generate Pareto sets for mulitiobjective problems
- All the above can be done dynamically

Reasons to use GBML 2

- Adapt population dynamically
  - to improve accuracy
  - to deal with non-stationarity
  - to minimise population size
    - to reduce overfitting
    - to improve run-time
    - to improve human-readability
- GBML's accuracy may not suffer from epistasis as much greedy search ([99] §12.1.1)
- Evolution can be used as a wrapper for any learner
  - The approach is universal
- Population-based search is naturally suited to parallel implementation

Reasons to use GBML 3

- [219] From an optimisation perspective, learning problems are typically:
    - Large
    - Non-differentiable
    - Noisy
    - Epistatic
    - Deceptive
    - Multimodal
- To which we can add:
    - High-dimensional
    - Highly constrained
- EAs are a good choice for such problems
- See [64] and §11 for more

Reasons against using GBML

- Algorithms typically more complex
  - Harder to implement
  - Harder to analyse
  - Less theory to guide development of new algorithms
- Increased run-time
- Not always appropriate
  - Run-time may be prohibitive
  - Same for set-up time
  - Simpler/faster methods may suffice
  - Improvements may be marginal
  - Bias of a given GBML method may be inappropriate for a given problem
    - In other words: it may not work well!
- See also SWOT (Strengths, Weaknesses, Opportunities, Threats) analysis of GBML [236]

Reading

General overviews of GBML

- Goldberg's classic 1989 text [113]
- The Hitch-Hiker's Guide to Evolutionary Computation is sadly no longer being updated but is still a valuable resource [126]
- Freitas' excellent 2002 book [99]

Contents

Phenotypic complexity and plasticity

Terms:

- Genotype: an individual's genes
- Phenotype: an individual's body (built based on genes)

Evolution can output a huge range of phenotypes

- From scalar values to complex learning agents

Agents can be more or less plastic (able to adapt)

- A fixed hypothesis does not learn
- A neural net with backprop can learn much, e.g.
  - Evolution specifies network structure and/or learning algorithm
  - But backprop adapts network weights

Contents

Classifying GBML systems by role

Categories:

- Evolutionary optimisation for sub-problems of learning
- GBML as learning
- GBML as meta-learning

- The output of learning is a fixed hypothesis
- When evolution adapts hypotheses, it is the learner
- When evolution adapts learners, it is a meta-learner

Evolutionary optimisation for sub-problems of learning

- Feature selection
  - Which features should the learner use as input?
- Feature construction
  - Can we combine existing feature to make more informative ones?
- Other uses of evolutionary optimisation within learning agents
  - Not many, but some e.g.
    - selecting training inputs
    - optimising weights in weighted k-nearest neighbour algorithm
    - replacement for beam search in the AQ algorithm
    - a search method in Inductive Logic Programming

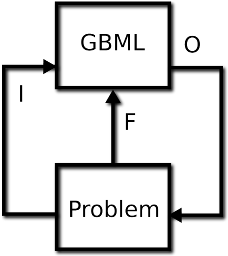
Structure of GBML systems

We can divide any evolutionary (meta)-learning system into parts:

- Representation:
  - Genotype: learner's genes
  - Phenotype: learner, built according to genes
    - In simple cases genotype and phenotype may be identical e.g. ternary LCS rules
- Feedback:
  - Learner's objective function (e.g. error function in SL)
  - Evolution's fitness function
- Production system: applies the phenotype to the problem
- Evolutionary system: adapts the genes

GBML as learning

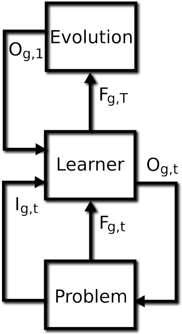
- GBML can evolve simple predictors which learn little or nothing themselves



Input **O**utput and **F**itness shown

GBML as meta-learning

- *Universal*: any learner can be augmented by GBML



- The learner (or a set of learners) is the output of evolution
- Subscripts denote **g**eneration and **t**ime step $(1 \ldots T)$

Meta-learning

- Meta-learning: learning about learning
- A broad term with different interpretations
- A meta-learner may:
  - optimise parameters of a learner
  - learn which learner to apply to a given input or a given problem
  - learn which representation(s) to use
  - discover update rules used to train learners
  - learn an algorithm which solves the problem
  - evolve an ecosystem of learners
  - potentially be open-ended
- See [300, 112] on non-evolutionary meta-learning
- Hyperheuristcs are another approach [46, 170, 171, 45]
  - 'Heuristics to learn heuristics'
  - A subset are evolutionary

Contents

Classifying GBML systems algorithmically

Pittsburgh (Pitt) approach:

- 1 chromosome = 1 solution
- Fitness assigned to complete solution
- Credit assignment problem:
  - How did genes contribute to fitness of chromosome?
  - Left to EA to deal with

Michigan approach:

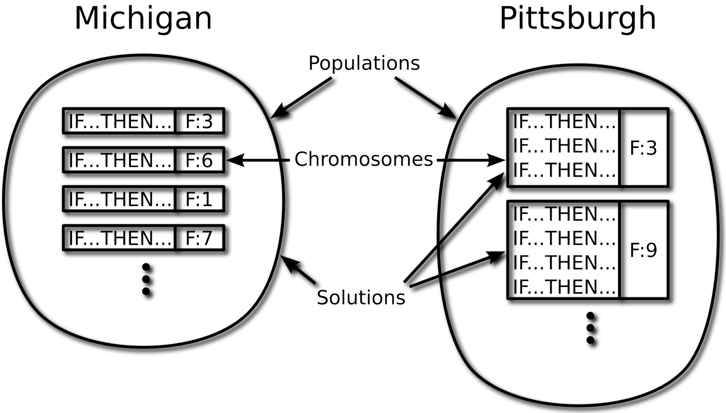
- 1 solution = many chromosomes
- Fitness assigned to partial solutions
- Credit assignment problem:
  - Chromosomes compete, complement and cooperate
  - How to encourage coverage of inputs, complementarity and cooperation?
  - How to measure a chromosome's contributions to solution (i.e. its fitness)?

Some hybrids exist e.g. [309]

Examples

LCS are rule-based systems

- Pitt LCS: chromosome is a variable-length *set* of rules
- Michigan LCS: chromosome is a fixed-length rule



The F:x associated with each chromosome indicates its fitness.

Pitt and Michigan compared

Pittsburgh:

- Slower
  - They evolve more complex structures
  - They assign credit at a less specific level
  - This is less informative
  - But see [9] and the slide on windowing
- Less complex credit assignment / more robust
- Since chromosomes are more complex so are genetic operators

Pitt and Michigan compared

Michigan:

- Finer grain of credit assignment than Pittsburgh approach
- Bad partial solutions can be deleted without restarting from scratch
  - More efficient
  - Also more suitable for incremental learning
- However: credit assignment is more complex
  - Solution is a set of chromosomes:
    - population must not converge fully
    - best set of chromosomes $\neq$ set of best chromosomes
- Mainly used in LCS

See [115, 149, 309, 100, 162] for comparisons

Michigan vs. Pitt: training

Pitt:

- Typically algorithm-driven
- Typically offline

Michigan:

- Typically data-driven
- Typically online
- More often used as learner for Reinforcement Learning (RL)
  - RL is almost always on-line
  - Not necessarily more often used a *meta-learner* for RL

Michigan production system

On each time step:

1. Identify action set: subset of population which match current input
2. Compute support in match set for each class
3. Select class o
4. Identify action set: subset of match set which advocates selected class
5. Update action set based on error
6. Optionally alter population

Iterative Rule Learning (IRL)

- A variation on Michigan approach
- 1 solution = many chromosomes
- But only 1 best chromosome selected after each run
  - Alters co-evolutionary dynamics
- Output of multiple runs combined
- Originated with SIA (Supervised Inductive Algorithm) [299, 200]
  - A supervised genetic rule learner

Genetic Cooperative-Competitive Learning (GCCL)

- A Michigan approach
- On each generation:
  - A new population is produced genetically and ranked by fitness
  - A 'coverage-based filter' allocates inputs to the first rule which correctly covers them
    - inputs are only allocated to one rule per generation
    - rules which have no inputs allocated die at end of generation
  - The remaining rules' collective accuracy is compared to the previous best generation (stored offline)
    - If new generation is more accurate (or the same but has fewer rules) it replaces the previous best
- Examples include COGIN [115, 116], REGAL [109] and LOGENPRO [323]

Contents

Evolution and learning as global and local search

Global search

- Good at finding a good basin of attraction
- Bad at finding optimum
- EAs are generally global

Local search:

- Opposite of above
- Learning methods are often local

We can get the best of both [327]:

- *Memetic* algorithms combine global and local search [123, 124, 227, 225, 264, 226, 252]
  - See [169] for a self-contained tutorial
- Generally outperform either alone
- E.g. evolve initial NN weights, then train with gradient descent
- 2 orders of magnitude faster than random initial weights [96]

Darwinian and Lamarckian evolution

Lamarckian Evolution/Inheritance

- Learning directly alters genes passed to offspring
- Offspring inherit the result of learning
- Does not occur in nature but can in computers
- Possibly more efficient than Darwinian evolution since result of learning not thrown away
  - [2] showed Lamarckian evolution much faster on stationary learning tasks
  - However, [256] showed Darwinian evolution generally better on non-stationary tasks
  - See also [308, 326, 240, 305]
- See [119] for a Lamarckian LCS

Baldwin effect: smoothing

Baldwin effect I: smoothing fitness landscape

- Phenotypic Plasticity: the ability to adapt (e.g. learn) during lifetime
- Suppose a mutation would have no benefit except for PP
- Without PP mutation does not increase fitness
- With PP mutation increases fitness
- Thus PP helps evolution (smooths fitness landscape)

Possible example: adult lactose tolerance

- Mutation allows adult humans to digest milk
- Humans learn to keep animals for milk
- . . . which makes mutation more likely to spread

Baldwin effect: smoothing

- Smoothing effect depends on PP
- The greater the PP the more potential for smoothing
- ALL GBML methods exploit BE to the extent they have PP
- See ([305] §7.2) for short review of BE in Reinforcement Learning

Baldwin effect: assimilation

Baldwin effect II: genetic assimilation

- Suppose PP has a cost (e.g. learning involves making mistakes)
- If PP can be replaced by new genes, it will
- E.g. a learned behaviour becomes instinctive
- Allows learned behaviours to become inherited without Lamarckian inheritance

Baldwin effect: bias

Baldwin effect and bias [293]

- All inductive algorithms have a bias
- Baldwin effect can be seen as shift from weak to strong bias
- Weak bias = learning
- Strong bias = instinctive behaviour

Contents

Online evolutionary computation

- In many problems (esp. sequential ones) feedback is very noisy and needs averaging
- [305] allocate trials to chromosomes in proportion to their fitness
  - At new generation evaluate each chrom. once
  - Allocate subsequent evaluations using softmax distribution
  - Recalculate average fitness after each evaluation
  - In non-stationary problems use recency-weighted average
  - They call this *online EC*
- Less time is wasted evaluating weaker chromosomes
- In online learning (where mistakes matter), fewer mistakes made
  - However, only on average; worst-case not improved
- Related to other work on optimising noisy fitness functions [274, 19], but they do not reduce online mistakes

Steady-state EAs

- Generational EAs evaluate entire population before replacing any
- Steady-state EAs [97] evaluate only a (typically small) proportion
  - E.g. in XCS only 2 individuals created and 2 deleted
  - Allows best individuals to reproduce immediately
  - Removes worst individuals more quickly
  - Less disruptive than generational
  - In online learning immediately improves population and hence decision making
- Applies selective pressure at two points:
  - Reproduction
  - Deletion

Co-evolving learners and problems

- Evolve learners **and** problems
- Learners can gradually solve harder problems
- We can discover what kinds of problems are hard or easy for a learner
- We can explore dynamics between them

Contents

GBML areas

Notes

- This section organised by phenotype and research community
- Communities are more disjoint than methods

Areas

- Sub-problems of learning
- Genetic Programming
- Evolving ensembles
- Evolving neural networks
- Evolving rule-based systems
    - Learning Classifier Systems
    - Genetic Fuzzy Systems

Contents

Evolutionary feature selection

- Some input attributes (features) contribute little or nothing
- We can simplify and speed learning by selecting only useful ones
    - EAs are widely used in the wrapper approach [151]
        - Learner treated as a black box optimised by search algorithm
    - Usually give good results compared to non-evolutionary methods [147, 262, 174] but there are exceptions [147]
    - EDAs found to give similar accuracy but run more slowly than a GA [65]
- More generally we can weight features (instead of all-or-nothing selection)
    - Some learners use weights directly e.g. weighted k-nearest neighbours [247]
- See [279, 11] for recent real-world applications
- Evolutionary methods are slower than non-evolutionary ones
- See [212, 99, 100] for overviews

Evolutionary feature construction

- Some features not very useful by themselves, but can be when combined with others
  - We can leave base learner to discover this itself
  - Or we can preprocess data to construct informative features
  - E.g. new feature $f_{\text{new}} = f_1$ AND f_3 AND f_8
  - Also called constructive induction
- Using GP to construct features out of the original attributes e.g. [139, 172, 265]
- Linear feature transformation by evolving a vector of coefficients [153, 245]
- Simultaneous feature transformation and selection had good results [247]

Other sub-problems of learning

- Training set optimisation:
  - Selecting training inputs [145]
  - Generating synthetic inputs [333, 72]
  - Partitioning data into training sets [250]
- Optimisation within a learner e.g.
  - Weighted k-nearest neighbours optimised with a GA [153]
  - Optimisation of decision tree tests using a GA and Evolutionary Strategy [64]
  - Optimisation of voting weights in an ensemble [285, 286]
  - [149] replaced beam search in AQ with a genetic algorithm
  - Inductive Logic Programming driven by a GA [282, 86, 84, 85, 283]
- Rule extraction
  - Extracting rules from NN e.g. [255, 209]
- Fitness function approximation
  - No known evolutionary examples but see [221] which used backprop

Contents

Genetic Programming

- A major evolutionary paradigm which evolves programs [298]
- Difference between GP & GA is not precise but typically GP:
  - evolves variable-length structures, most commonly trees
  - genes/nodes can be functions
- Usually Pittsburgh
- We cover 2 representations:
  - GP trees
  - decision trees
  - see also [325]

GP and GAs compared

- Following differences arise because GP representations are more complex

Pros of GP:

- Easier to represent complex languages e.g. first-order logic
- Easier to represent complex concepts compactly
- GP is good at finding novel, complex patterns overlooked by other methods. See ([99] §7.6)

Cons of GP:

- Expressive representations have large search spaces
- GP tends to overfit / does not generalise well
- Variable-length representations have problems with bloat (see e.g. [244])

GP for learning

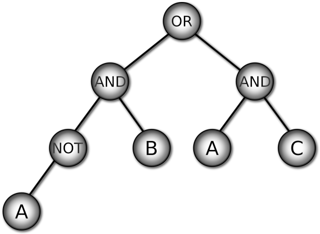
- GAs typically applied to function optimisation
- GP widely applied to learning
- Koza defined a set of 'typical GP problems' [167]
- More-or-less agreed benchmarks for GP community [298]
- They include:
    - Multiplexer and Parity Boolean functions
    - Symbolic regression of mathematical functions
    - The *Intertwined Spirals* problem: classification of 2D points as belonging to one of two spirals
- All the above are more naturally posed as learning than optimisation

GP trees for classification

To classify an input:

- Instantiate leaf variables with input's values
- Propagate values upwards from leaves though functions in non-leaf nodes
- Output is the value of the root (top) node

Attribute A	Attribute B	Attribute C	Class
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

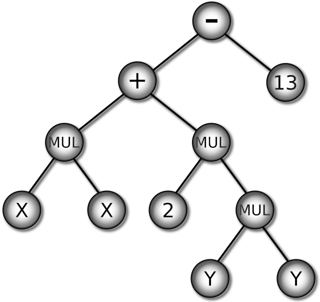


GP trees for regression

In regression problems:

- leaves may be constants
- non-leaves are mathematical functions

$$x^2 + 2y^2 - 13$$

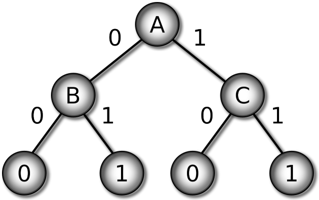


Decision trees

To classify an input:

- start at root (top) of tree
- follow branch corresponding to value of attribute in node
- repeat until leaf reached
- value of leaf is classification of input

Attribute			Class
A	**B**	**C**	
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



Evolving decision trees

Basic approach:

- leaf nodes are classes
- non-leaf nodes are tests of attributes
- branches are attribute values
- fitness is accuracy of classification on training set

Evolving first-order and oblique decision trees

First-order trees: [251]

- Uses both propositional and first-order internal nodes
  - First-order logic makes trees more expressive
  - Allows much smaller solutions than found by CN2 (a rule learner) or C4.5 (tree learner)
  - Accuracy similar

Oblique (linear) trees: [31]

- Conventional tree algorithms learn axis-parallel decision boundaries
- Oblique trees make tests on a linear combination of attributes
- More expressive but larger search space

Evolving individual nodes in DTs

- In most GP-based tree evolvers an individual is a complete tree
- In [210] each individual is a tree node
- Tree is built incrementally
  - 1 GP run is made for each node
  - Like IRL, but results are added to a tree structure, not a list
- Results:
  - Non-leaf nodes (and hence trees) are more complex than usual
  - Trees are somewhat easier to understand as nodes can be analysed separately

Ensemble methods and GP

Ensemble ideas have been used in different ways

- To reduce fitness computation time and memory requirements
  - Training on subsamples of the data
    - Bagging approach: [98, 143]
    - Boosting approach: [272]
- To improve accuracy using an ensemble of GP trees [156, 239]
  - Each run adds one tree to ensemble
  - Weights computed with standard Boosting

Limited Error Fitness

- [105] introduced LEF
- A way of reducing run-time
- Proportion of training set used to evaluate fitness depends on individual's performance
- No test set used in [105] but one could be

GP Hyperheuristics

Ways of expanding the power of evolutionary search

- [259] proposes a meta-GP system which evolves evolutionary operators
- [99] (§12.2.3) sketches an approach to 'algorithm induction'
  - Instead of evolving decision rules GP evolves classification algorithms
  - [238] is a book devoted to this subject
- [44] discusses GP hyperheuristics

Lack of test sets in GP

GP terminology:

- Follows convention in GA field since at least [130]
- Brittleness: overfitting; poor generalisation to unseen cases
- Robustness: good generalisation

Evaluation:

- GP usually evaluated only on training set [176, 298]
- Sometimes test set used inappropriately [176]
- Nonetheless has same need for test sets as other methods [176]

Inductive generalisation:

- One of the open issues for GP identified in [298]
- See [176, 298] for various methods for encouraging generalisation in GP

Research directions

- Hyperheuristics
- Generalisation to test sets

Reading

- Koza's 1994 book [168] for the basics of evolving decision trees with GP
- Wong and Leung's 2000 book on data mining with grammar-based GP [323]
- Freitas' 2002 book [99] for a good introduction to GP, decision trees and evolutionary and non-evolutionary learning
- Poli, Langdon and McPhee's free 2008 GP book [244]
- Vanneschi and Poli's 2010 survey of GP [298]
- The GP Bibliography has over 5000 entries [179]

Contents

Ensembles

- Also called 'Multiple Classifier Systems' and 'Committee Machines'
- The field which studies how to combine predictions from multiple sources
  - Widely applicable to evolutionary systems where a population provides multiple predictors
  - But can be used with any learning method
  - Although most useful for unstable learners
  - Can be heterogeneous (composed of different types of predictors); called *hybrid* ensembles
    - Few hybrid studies exist [36] but see e.g. [324, 74, 71]
- Some good theoretical foundations [36, 292]
- Identified by Dietterich as 1 of 4 current directions for Machine Learning in 1998 [83]

Ensembles 2

- Issues
  - How to create or select ensemble members?
  - How many members are needed?
  - When to remove ensemble members?
  - How to combine their predictions?
  - How to encourage diversity in members?
- Key advantage: better test set generalisation [66]
- Other advantages [261]
  - Can perform more complex tasks than individual members
  - Overall system can be easier to understand and modify
  - More robust / graceful degradation
- Many approaches
  - Best known are bagging and boosting

Unstable learners

Ensembles are most effective with unstable learners:

- Their hypotheses are sensitive to various parameters
- Allows construction of an ensemble with diverse errors
- Effectively, learners whose bias can be altered e.g.
  - by random initialisation of NN weights
  - by sampling data differently for each predictor
  - by weighting data according to errors made by other predictors
  - by altering features used
  - by altering representations used
- Unstable learners: decision trees, Radial Basis Function networks, evolutionary meta-learning ...
- Stable learners: majority class prediction, Support Vector Machines ...

Ensembles are multiobjective

Ensembles exploit diversity in predictors

- Multiple identical predictors provide no advantage
- But an ensemble of predictors making different errors is useful
- Combine predictions so that ensemble output is at least as good on training set as average predictor [173]
- We want accurate predictors with diverse errors [83, 122, 173, 229, 228]
- Hence a multi-objective problem [294]

In addition we may want to minimise ensemble size

- Reduces run time
- Can make ensemble easier to understand
- Evolving variable-length chromosomes results in bloat

Diversity from bagging and boosting

- Families of well-known methods for training ensembles
- Bagging: [32, 33]
  - Generate training subsets by sampling uniformly with replacement
  - Each classifier trains on a different subset
- Boosting (and leveraging): [101, 102, 214]
  - Allocate training data to each classifier in sequence
  - First classifier samples data uniformly
  - Later classifiers more likely to sample data misclassified earlier
- Effects:
  - Increases their diversity
  - Alters their bias

Evolutionary ensembles

- Most ensembles are non-evolutionary
- But evolution has many applications
  - Classifier creation and adaptation
    - Provides ensemble with set of candidates
  - Voting
    - [177, 285, 286, 73] evolve weights for the votes of ensemble members
  - Classifier selection
    - Winners of evolutionary competition added to ensemble
  - Feature selection
    - Generate diverse classifiers by training them on different features
    - See §1 and ([175] §8.1.4)
  - Data selection
    - Generate diverse classifiers by training on different data
    - See §1
- All have non-evolutionary alternatives

Classifier creation and adaptation

- Single vs. multi-objective
  - Single-objective evolution common e.g. [201]
    - Fitness combines accuracy and diversity into a single objective
  - Evolutionary multiobjective optimisation is an active area
    - Can upgrade GBML to multi-objective GBML
    - Multi-objective evolutionary ensembles are rare [71]
    - But starting to appear e.g. [1, 71, 70]
- Other measures to evolve diversity
  - Fitness sharing e.g. [202]
  - EEL's co-evolutionary fitness [104]

Evolutionary Ensemble Learning (EEL) [104]

- Compares boosting and co-evolution of learners and problems
  - Both gradually focus on cases which are harder to learn
  - Argues co-evolution less likely to overfit noise
- Introduces co-evolution inspired fitness
  - Let Q be a set of reference classifiers
  - Hardness of a training example x_i based on how many members of Q misclassify it
  - Fitness of a classifier sum of hardnesses of x_i it classifies correctly
  - Q is the population of classifiers
  - Results in accurate yet diverse classifiers
- Introduces greedy margin-based selection of ensemble members
- Simpler off-line version dominates on-line version
  - On-line version lacks a way to remove bad classifiers
- Good results compared to Adaboost on 6 UCI [8] datasets

Evolutionary selection of members

Two extremes:

- Usually each run produces 1 member
  - Many runs needed
- Sometimes entire population eligible for ensemble
  - Only 1 run needed

Latter does not resolve selection problem:

- Which to use?
- Many combinations possible!
- Set of best individuals may not be best ensemble
- Formally equivalent to feature selection problem ([104] §3.2)
- See e.g. [263, 253] for evolutionary approaches

Research directions

- Multi-objective ensembles [71]
- Hybrid ensembles [71]
- Minimising ensemble complexity [202]

Reading

- Opitz and Shavlik's classic 1996 paper on evolving NN ensembles [229]
- Kuncheva's 2004 book on ensembles [175]
- Chandra and Yao's 2006 [70] discussion of multi-objective evolution of ensembles
- Yao and Islam's 2008 review of evolving NN ensembles [328]
- Brown's 2005 and 2010 surveys of ensembles [36, 34]

Contents

Artificial Neural Networks

- A NN consists of:
  - A set of nodes (input, output and hidden)
  - A set of directed connections between nodes
    - Connections specify inputs and outputs to nodes
  - A set of weights on the connections
- Nodes compute by:
  - Integrating their inputs using an activation function
  - Passing on their activation as output
- NNs compute by:
  - Accepting external inputs at input nodes
  - Delivering outputs to output nodes

Evolving neural networks

Acronyms include

- EANNs (Evolving Artificial Neural Networks) [327]
- ECoSs (Evolving Connectionist Systems) [155]

Evolution has been applied at 3 levels:

- Weights
- Architecture
  - connectivity: which nodes are connected
  - activation functions: how nodes compute outputs
  - plasticity: which nodes can be updated
- Learning rules

Representations

- Direct encoding [327, 96]
  - all details (connections and nodes) specified
- Indirect encoding [327, 96]
  - only key details (e.g. number of hidden layers and nodes)
  - a learning process determines the rest
- Developmental encoding [96]
  - a developmental process is genetically encoded [157, 120, 222, 142, 237, 281]

Uses:

- Indirect and developmental representations are more flexible
  - tend to be used for evolving architectures
- Direct representations tend to be used for evolving weights alone

Credit assignment

- Virtually always Pittsburgh approach
- A few Michigan systems: [5, 268, 271]

Michigan: each chromosome specifies only one hidden node

- How to define architecture?
  - Simple method: fix architecture
- How to make nodes specialise?
  - Encourage diversity during evolution: e.g. fitness sharing
  - Increase diversity after evolution: prune redundant nodes [5]

Two ways of adapting weights

Learning:

- Most NN learning algorithms are based on gradient descent
- Including the best known: backpropagation (BP)
- Many successful applications, but often get trapped in local minima [280, 306]
- Require a continuous and differentiable error function

Evolving:

- EAs don't rely on gradients and can work on discrete fitness functions
- Much research has been done on evolution of weights

Evolving NN weights

- Fitness functions typically penalise: NN error and complexity (number of hidden nodes)
- The expressive power of a NN depends on the number of hidden nodes
- Fewer nodes = less expressive = fits training data less
- More nodes = more expressive = fits data better
- Too few nodes: NN underfits data
- Too many nodes: NN overfits data

Evolving weights vs. gradient descent

Evolution has advantages [327]:

- Does not require continuous differentiable functions
- Same method can be used for different types of network (feedforward, recurrent, higher order)

Which is faster?

- No clear winner overall – depends on problem [327]
- Evolving weights AND architecture is better than weights alone (we'll see why later)
- Evolution better for Reinforcement Learning and recurrent networks [327]
- [96] suggests evolution is better for dynamic networks

Happily we don't have to choose between them . . .

Evolving AND learning weights

Evolution:

- good at finding a good basin of attraction
- bad at finding optimum

Gradient descent:

- Opposite of above

To get the best of both: [327]

- Evolve initial weights, then train with gradient descent
- 2 orders of magnitude faster than random initial weights [96]

Evolving NN architectures

- Arch. has important impact on results: can determine whether NN under- or over-fits
- Designing by hand is a tedious, expert trial-and-error process

Alternative 1:

- Constructive NN grow from a minimal network
- Destructive NN shrink from a maximal network
- Both can get stuck in local optima and can only generate certain architectures [6]

Alternative 2:

- Evolve them!

Reasons EAs are suitable for architecture search space

1. "The surface is infinitely large since the number of possible nodes and connections is unbounded
2. the surface is nondifferentiable since changes in the number of nodes or connections are discrete and can have a discontinuous effect on EANN's performance
3. the surface is complex and noisy since the mapping from an architecture to its performance is indirect, strongly epistatic, and dependent on the evaluation method used;
4. the surface is deceptive since similar architectures may have quite different performance;
5. the surface is multimodal since different architectures may have similar performance." [219]

Reasons to evolve architectures and weights simultaneously

Learning with gradient descent:

- Many-to-1 mapping from NN genotypes to phenotypes [329]
  - Random initial weights and stochastic learning lead to different results
  - Result is noisy fitness evaluations
  - Averaging needed – slow

Evolving arch. and weights simultaneously:

- 1-to-1 genotype to phenotype mapping avoids above problem
- Result: faster learning
- Can co-optimise other parameters of the network: [96]
  - [20] found best networks had very high learning rate
  - May have been optimal due to many factors: initial weights, training order, amount of training

Evolving learning rules [327]

There's no one best learning rule for all architectures or problems

- Selecting rules by hand is difficult
- If we evolve the architecture (and even problem) then we don't know what it will be a priori

Solution: evolve the learning rule

- Note: training architectures and problems must represent the test set
  - To get general rules: train on general problems/architectures, not just one kind
  - To get rule for a specific arch./problem type, just train on that

Evolving learning rule parameters [327]

- E.g. learning rate and momentum in backpropagation
- Adapts standard learning rule to arch./problem at hand
- Non-evolutionary methods of adapting them also exist
- [68] found evolving architecture, initial weights and rule parameters together as good or better than evolving only first two or third (for multi-layer perceptrons)

Evolving learning rules [327, 246]

- Open-ended evolution of rules initially considered impractical
- Instead generic update rule is given and its parameters evolved [69]
    - Generic update is a linear function of 10 terms
    - 4 terms represent local information about node being updated
    - 6 terms are the pairwise products of the first 4
    - The weight on each term is evolved as a vector of reals
    - Can outperform human-designed rules e.g. [81]
- Later GP used to evolve novel rule types [246]
    - GP used a set of mathematical functions
    - Result consistently outperformed standard BP
- Whereas architectures are fixed, rules could change over lifetime (e.g. learning rate)
    - But evolving dynamic rules is more complex

Ensembles of NNs

- Most methods output a single NN [328]
  - E.g. EPNet [329]
- However, evolving NNs are naturally treated as an ensemble
  - Population = ensemble
  - Recent work beginning to focus on evolving ensembles of NNs
- Evolving NNs is inherently multiobjective
  - We want accurate yet simple and diverse networks
  - Some work combines objectives into 1 fitness function
  - Others are explicitly multi-objective

Single-objective ensembles

- [330] used EPNet's population as an ensemble
  - Evolution (EPNet) was not modified
  - Result outperformed population's best individual
- [201] pursue accuracy and diversity in 2 ways:
  - Modify backprop to minimise error and maximise diversity
    - Called Negative Correlation Learning (NCL)
    - Errors of members become negatively correlated (diverse)
  - Fitness combines accuracy and diversity in a single objective

Single-objective ensembles 2

- EENCL (Evolutionary Ensembles for NCL) [202]
  - Automatically determines the size of an ensemble
  - Encourages diversity with fitness sharing and NC learning
  - Problem: how to combine many candidates into 1 ensemble?
    - many combinations possible!
  - Solution: cluster then select (see [150])
    - cluster candidates based on errors on training set
    - clusters make similar errors
    - most accurate in each cluster joins ensemble
  - Ensemble can be much smaller than the population
- CNNE (Cooperative Neural Net Ensembles) [146]
  - Used a constructive approach to determine
    - number of individuals
    - how many hidden nodes each has
  - Both contribute to expressive power of ensemble
  - Able to balance the two to obtain suitable ensemble
  - More complex problems needed larger ensembles

Multi-objective ensembles

- MPANN (Memetic Pareto Artificial NN) [1]
  - First use of multi-objective evolution for NNs
  - Uses gradient-based local search to optimise network complexity and error
- DIVACE (diverse and accurate ensembles) [70]
  - Multiobjective evolution maximises accuracy and diversity
  - Selection based on non-dominated sorting [273]
  - Clustering used to select ensemble members
  - Uses a variant of differential evolution [278] and simulated annealing

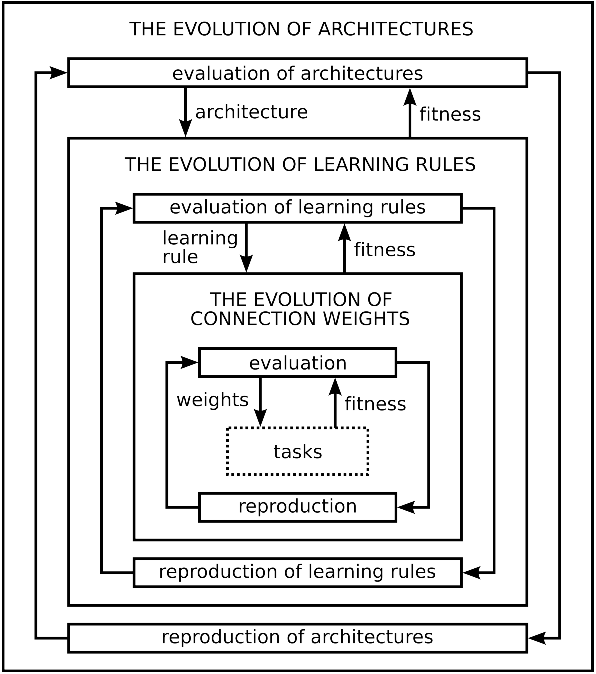
DIVACE-II

DIVACE-II (diverse and accurate ensembles) [71]

- A heterogeneous multiobjective Michigan approach
  - Role of crossover/mutation played by boosting and bagging (BB)
  - BB produces accurate and diverse candidates
  - NNs, Support Vector Machines and Radial Basis Function networks used
  - Only dominated members are replaced
- Each generation BB makes candidate ensemble members
- Performance
  - Very good compared to 25 other learners on Australian credit card and diabetes datasets
  - Outperforms DIVACE

Yao's framework for evolving NNs [327]

- Architectures, rules and weights can evolve as nested processes
- Weight evolution is innermost (fastest time scale)
- Either rules or architectures are outermost
  - If we have prior knowledge, or are interested in a specific class of either, this constrains search space
  - Outermost should be the one which constrains search space most
- Can be thought of as 3D space of evolutionary NNs where 0 on each axis represents one-shot search and infinity represents exhaustive search
- If we remove references to EAs and NNs it becomes a general framework for adaptive systems


THE EVOLUTION OF ARCHITECTURES
evaluation of architectures
architecture
fitness
THE EVOLUTION OF LEARNING RULES
evaluation of learning rules
learning rule
fitness
THE EVOLUTION OF CONNECTION WEIGHTS
evaluation
weights
fitness
tasks
reproduction
reproduction of learning rules
reproduction of architectures

Evolving NNs – conclusions [96]

- Most studies of neural robots in real environments use some form of evolution
- Evolving NNs can be used to study "brain development and dynamics because it can encompass multiple temporal and spatial scales along which an organism evolves, such as genetic, developmental, learning, and behavioral phenomena."
- "The possibility to co-evolve both the neural system and the morphological properties of agents . . . adds an additional valuable perspective to the evolutionary approach that cannot be matched by any other approach." p. 59

Reading

Reading on evolving NNs:

- Yao's classic 1999 survey [327]
- Kasabov's 2007 book [155]
- Floreano et al.'s 2008 survey [96]
  - includes evolving dynamic and neuromodulatory NNs
- Yao and Islam's 2008 survey of evolving NN ensembles [328]

Contents

Rule-based systems

- We distinguish two areas:
  - Learning Classifier Systems
  - Genetic Fuzzy Systems
- The two overlap:
  - GFS evolve fuzzy rules
  - Some LCS evolve fuzzy rules

Learning Classifier Systems

Background:

- Originated in GA community as a way of applying GAs to learning problems
- Terminology: CS, LCS, GBML
  - (L)CS sometimes taken to mean Michigan systems (see e.g. [115])
  - However it now generally includes Pitt systems (as implied by the "IWLCS" workshop and its contents)
  - Difficulty in naming [126] due in part to difficulty in defining what they are [266, 132]
- Evolve populations of condition/action rules called classifiers

Representations:

- Rules have limited expressive power
- A solution requires many rules; solutions are piecewise

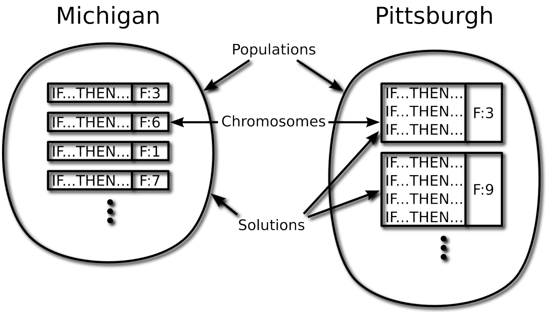
Production systems

- Pitt, Michigan, IRL and GCCL all used
- Michigan is rare elsewhere, but most common form of LCS
- IRL most common with Genetic Fuzzy Systems (but see [3] for a non-fuzzy version)

Michigan vs. Pitt: representations

Rule-based representations:

- Michigan: chromosome is **1** fixed-length rule
  - e.g. XCS
- Pitt: chromosome is a variable-length **set** of rules
  - e.g. GAssist



Michigan vs. Pitt: learning and evolution

Typically:

- Rule conditions and actions are evolved
- Phenotypic parameters are learned
  - Michigan: for each rule
  - Pitt: for each ruleset
- Examples:
  - UCS (Michigan SL) parameters:
    - Fitness
    - Mean action set size – for deletion which balances set sizes
    - Experience – to give confidence in fitness
  - GAssist (Pitt SL) parameters:
    - Fitness

LCS variations

Sometimes rules:

- predict next state
- read and write to memory
- are generated non-genetically

LCS: Representations

Ternary conditions

- Strings: (see e.g. [313])
  - All fixed length
  - Inputs are binary
  - Rules have 1 binary action and 1 ternary condition from $\{0, 1, \#\}$
  - $\#$ is a wildcard, matching 0 and 1 in inputs
  - E.g. 00$\#$ matches 000 and 001
- Very widely used, especially before ˜2000
- Inherited from GAs
  - Minimal alphabets
  - Parallel with binary GA schemata
- Limited expressive power [260] (but see also [27])
  - A factor in pathological credit assignment (strong/fit overgenerals [162])
- Various extensions studied

Real-valued interval conditions

Following [12] we distinguish 2 approaches

Representations based on discretisation:

- HIDER* uses "natural coding" [111]
- ECL clusters attribute values and evolves constraints on them [85]
- "Adaptive discretisation intervals" in GAssist [12]

Representations handling real values directly:

- HIDER (unlike HIDER*): genes specify a lower and upper bound (lower always $<$ upper) [3]
- Variation on HIDER: when upper $<$ lower, attribute is irrelevant [76]
- Intervals [310, 277]
- XCSR: genes specify a center and spread [315]

Default rules

- Should increase number of solutions without increasing basic search space
- Should allow gradual refinement of knowledge by adding exceptions [133]
- Can reduce number of rules needed for solution

Truth table			
A	**B**	**C**	**Output**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Ternary Rules
0 0 # → 0
0 1 # → 1
1 # 0 → 0
1 # 1 → 1

Default Rule
0 0 # → 0
1 # 0 → 0
→ 1

Default rules: Pitt

Pitt systems:

- GABIL [152] and GAssist [12] use decision lists
  - Each rule is an exception to any following, overlapping ones
  - Conflict resolution trivial; based on order
  - No need to assign credit to each rule
- Pitt LCS often evolve default rules (e.g. last rule in list is fully general)
- GAssist enforces fully general last rule [12, 13]
  - Initialised with all possible last rules
  - Evolution selects best

Default rules: Michigan

- Called default hierarchies in Michigan LCS
  - Specific rules are exceptions to general rules
  - Attempts to bias conflict resolution according to specificity
- Problems [270]
  - Hard to evolve – rules need to cooperate
  - Unstable – introduce interdependence between rules
  - Complicate credit assignment, since exception rules must override defaults [311, 270]
  - Fewer #s doesn't mean a rule matches fewer inputs
  - Why must exceptions be more specific?
- Consequences
  - Not much interest since early 1990s (but see [297])
  - Not all Michigan LCS support them (e.g. neither ZCS nor XCS)
  - Should be revisited from ensembles perspective

Other representations for conditions

- VL_1 logic [218] as used in GIL [148]
- First-order logic [215, 216, 217]
- Decision lists [249] used in GABIL [152] and GAssist [12]
- Messy encoding [181]
- Ellipses [53] and hyperellipses [59]
- Hyperspheres [211]
- Convex hulls [197]
- Hyperplane coding [29, 28]
- Tile coding [186]
- GP trees [4, 182, 183]
  - GP to define Boolean networks [37]
- Support vectors [208]
- Edges of an Augmented Transition Network [178]

Other representations for conditions and actions

- Gene Expression Programming [321]
- Fuzzy rules (see [24, 128])
- Neural networks [269, 78, 271, 43, 224, 79, 138, 137]
- Evolved prototypes
  - One of the representations used with GALE [204, 203, 207]
  - Evolve prototypes, use k-nearest-neighbour for classification
  - Prototypes need not be fully specified
  - Also used in GAssist [12]
- Decision trees
  - Another of GALE's representations
  - Uses GP to evolve trees defining axis-parallel and oblique hyper-rectangles [207]
- Computed actions [291, 196]
- Continuous actions [320]

Evolutionary selection of representations

- There are a lot of representations to choose from!
- Which one to use for a problem?
  - Or each part of a problem
- Let evolution decide!
  - Helps adapt bias of learner
  - A form of meta-learning

Selecting default actions in decision lists

GAssist (Pittsburgh)

- Rulesets are decision lists
- Initialise rulesets with fully general last rule
    - A default action
- Evolution selects most suitable default action [12, 13]
- For good results need to encourage diversity in default actions

Selecting classification algorithms

GALE (Pittsburgh) [203]

- Has elements of Cellular Automata and Artificial Life:
  - Individuals distributed on a 2D grid
  - Only neighbours (within r hops) interact:
    - 2 neighbours can perform crossover
    - An individual can be cloned and copied to a neighbouring cell
    - An individual may die if its neighbours are fitter
- Representations:
  - Rule sets, prototypes, and decision trees (orthogonal, oblique, and multivariate based on nearest neighbor)
  - Population may be homogeneous or heterogeneous
  - In [207] GALE modified to interbreed orthogonal and oblique trees

Selecting condition shapes

Representational ecology [211]

- Two boolean classification tasks:
  - Plane function: easy to describe with hyperplanes, hard with hyperspheres
  - Sphere function: opposite
- 3 versions of XCS:
  - with hyperplane conditions (XCS-planes)
  - with hyperspheres (XCS-spheres)
  - with both (XCS-both)
- XCS otherwise unchanged
  - In XCS-both representations compete due to XCS's pressure against overlapping rules

Selecting condition shapes

- Planes and spheres do not interbreed
  - Genetically independent populations: species
- Classification accuracy:
  - XCS-planes: good at plane function, bad at sphere function
  - XCS-spheres: opposite
  - XCS-both: good at both
    - Selected the better representation for each task
    - No significant difference in accuracy compared to better single-representation version
- Learning speed of XCS-both:
  - Similar speed to XCS-sphere on sphere function
  - But significantly slower than XCS-plane on plane function

Selecting discretisation methods and cut points

Discretisation of real attributes in GAssist

- Adaptive Discretization Intervals (ADI) approach has 2 parts

Adapting interval sizes

- A discretisation alg. proposes cut points for each attribute
- This defines the finest discretisation possible: micro-intervals
- Evolution can merge and split macro-intervals, composed of micro-intervals
- Each individual can have different macro-intervals

Selecting discretisation algorithms

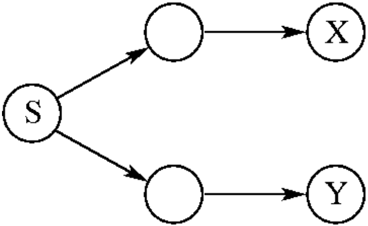
- Evolution can select discretisation algorithms for each attribute or rule
- Selects from a pool of algorithms including uniform-width, uniform-frequency, ID3, Fayyad & Irani, Màntaras, USD, ChiMerge and random
- Difficult to evolve the best discretisers

Models of the world

- A basic rule consists of a condition, action, and strength
- For modelling, add an *expecton* – a prediction of the next state
  - IF (0 1 #) THEN (take action 0 1) expect (state 0 1 #)
  - Called an *Anticipatory* classifier system
- Allows planning and latent learning (learning in the absence of reward) e.g. [248, 131, 275, 276, 49, 40, 107, 106, 331, 223, 38, 42].
- Learned with non-evolutionary methods (so we won't cover details)

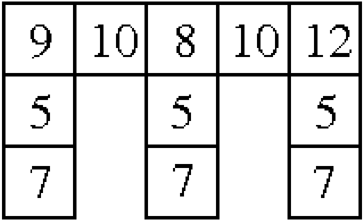
Latent learning

- Let the learner explore the maze without reward
- Place the learner in state X and reward it
- Place the learner in state S and see whether it goes to X or Y



Internal memory

- *Stimulus-response* LCS have no internal memory
- Memory is needed to solve tasks where inputs are *perceptually aliased*, e.g. McCallum's maze [213]

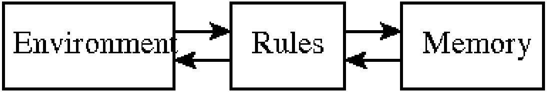


- States labelled with same number appear identical to learner

Evolving memory: message lists and bit registers

Ways of adding explicit memory:

- A message list [134] p.110
  - if (input = 001 and list contains 010) then (take action 0, post 111)
- Bit registers [312, 181, 180, 193]
  - if (input = 001 and bit 1 is set) then (take action 0, clear bit 1)



- Actions can add messages to the list / set bits in the register
- Conditions can match messages / register settings
- See also [91, 267, 75]

Evolving memory: corporations

- In between Michigan and Pitt approaches:
  - Selection occurs partly on groups (corporations of classifiers)
- Rules dynamically form corporations
  - A special genetic operator links rules in successive match sets
  - Rules in a corporation have collective fitness
  - Corporation is deleted or reproduced as a whole
  - However rules are updated and make predictions independently
  - Removes competition for reproduction
- Rules in corporation fire in sequence
  - This linkage provides a form of memory
- Proposed in [322], implemented in e.g. [267, 289, 288, 290]

Evolving memory: Augmented Transition Networks

ATN:

- Introduced as parsers for natural language
- A graph in which nodes represent states and edges transitions
- Transitions are non-deterministic
- Registers provide memory

ATNoSFERES: [178]

- Pitt LCS where rules are edges in an ATN
- Environment is "parsed" as a sentence would be
- Evolves automata which can solve problems needing memory
- Found better policies than other LCS, but run-time much longer
- Not clear whether non-determinism is helpful
  - In non-Markov problems deterministic policies can get stuck
  - However, non-deterministic policies are harder to evaluate

Macroclassifiers

- Optimisation for Michigan populations introduced in [313]
- As population converges on solution set, many identical copies accumulate
- Use 1 rule to represent many identical virtual rules [313]
  - Number of virtual copies called the *numerosity* of the rule
- Reduces runtime and provides interesting statistics
- Empirically, macroclassifiers perform essentially as the equivalent 'micro'classifiers [159]

Example of macroclassifiers

Without Macroclassifiers

Rule	Cond.	Action	Strength
m	##0011	1	200.0
m'	##0011	1	220.0
n	##0011	0	100.0
o	001110	1	100.0

With Macroclassifiers

Rule	Cond.	Action	Strength	Numerosity
m	##0011	1	200.0	2
n	##0011	0	100.0	1
o	001110	1	100.0	1

LCS: Rule Discovery

Evolution in Pitt and Michigan LCS

- Pitt:
  - *Multiple objectives.* Accuracy and parsimony of rulesets
- Michigan:
  - *Multiple objectives.* Coverage, accuracy, and parsimony of population
  - *Co-evolution.* Rules cooperate and compete
  - *Fitness sharing* to encourage diversity
  - *Crowding.* Deletion probability is proportional to degree of overlap with other rules
  - *Restricted mating.* See Niche GA

Windowing in Pitt LCS

- Pitt LCS are slower than Michigan
  - Naive approach: each individual evaluated on entire data set
- Windowing: learn on data subsets to improve runtime [103]
  - windowing has been used in Pitt LCS since at least ADAM [117] (see also description in [115] p.235)
- E.g. ILAS (Incremental Learning by Alternating Strata) [12]
  - Partition data into *n* strata with class distribution of entire set
  - Use a different stratum for each generation (iterate)
  - On larger data sets speed-up can be an order of magnitude
  - Other data sampling techniques applicable
- Side effect
  - Can reduce overfitting, improve test accuracy
  - Specific rules starved; fewer and more general rules evolve
- Tuning
  - Number of strata determines speed-up and over/underfitting
- Windowing is used as standard on recent real-world applications e.g. [14, 10, 9])

Michigan rule discovery

- Most rule discovery research has focused on Michigan LCS as:
    - Evolutionary dynamics are more complex
    - Michigan LCS are more common
- Rest of this section deals with Michigan systems
    - although many ideas could be applied to Pitt
    - e.g. self-adaptive mutation
- Unusual emphasis in Michigan LCS on minimising population size
- Various techniques:
    - Generalisation term in fitness
    - Subsumption deletion
    - Condensation
    - Compaction methods
- Michigan LCS use Steady State GAs which are useful for on-line learning

Niche GAs

- *Panmictic GA*: all rules eligible for reproduction
- *Niche GA*:
    - Mating restricted to rules in same action set (a 'niche')
    - Such rules' input spaces overlap and actions agree
        - They make related predictions
    - Mating related rules is more effective, on average
- Other effects: [313]
    - Strong bias towards general rules, since they match more
    - Pressure against overlapping rules, since they compete [162]
    - Complete coverage, since competition occurs for each input
- A form of speciation; creates non-interbreeding sub-populations
- Notes
    - Introduced in [26]
    - Original restriction was to match set
    - Further restricted to action set in [314]
    - Used in XCS and UCS
    - Related to "universal suffrage" in [110]

EDAs instead of GAs

- EDA: Estimation of Distribution Algorithm
    - A form of stochastic search
    - Like a GA, but no crossover or mutation
    - Instead it iteratively:
        - samples individuals from a probabilistic model
        - updates model based on their fitness
- [61, 60, 62] replaced XCS's usual crossover with EDA-based method to improve solving of difficult hierarchical problems
- [205, 206] introduced CCS: a Pitt LCS based on compact GAs (a simple form of EDA)

Subsumption deletion

- Introduced in XCS (see [52])
- When rule x subsumes rule y, y is deleted and the numerosity of x incremented
- In XCS a rule is allowed to subsume another if:
  - It logically subsumes it
  - It is accurate (has low prediction error)
  - It is experienced (has been evaluated sufficiently)
    - so we can be sure it is accurate
- A rule x *logically subsumes* a rule y when x matches a superset of the inputs y matches, and they have the same action
  - E.g., $00\#\rightarrow 0$ subsumes $000\rightarrow 0$ and $001\rightarrow 0$

Two checks for subsumption

GA Subsumption

- When child created, check to see if its parents subsume it
- Constrains accurate parents to only produce more general children

Action Set Subsumption

- Most general of the accurate, experienced rules in action set subsumes others
- Removes redundant specific rules from the population
- Too aggressive for some problems

Michigan evolutionary dynamics

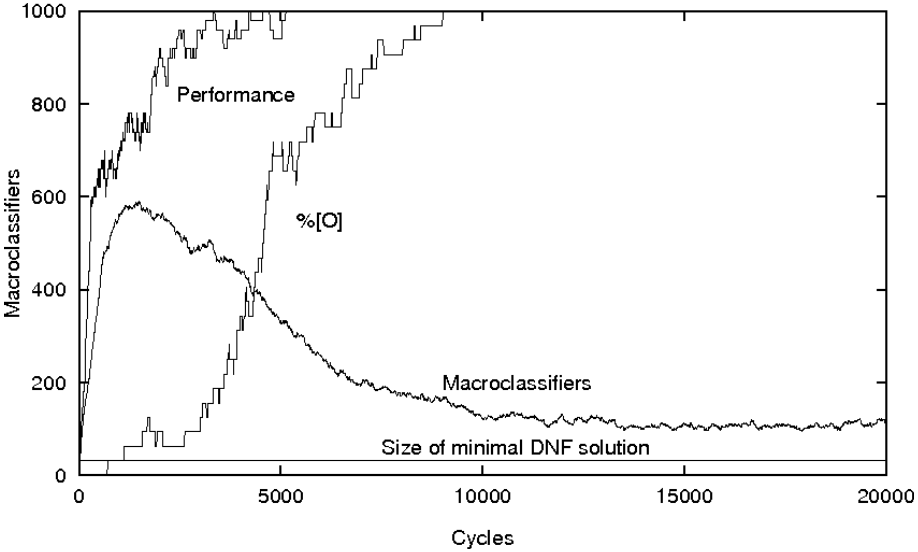
Next slide shows:

- XCS learning 11 multiplexer (Boolean function)
- Performance: moving average of accuracy
- Macroclassifiers: number of unique condition/action rules
- %[O]: proportion of minimal set of 16 ternary rules XCS needs to represent solution

Notes:

- Initial population empty; created by covering
- Population size limit 800
- All input/output pairs in train and test sets
- Curves average of 10 runs
- Other settings as in [313]

Evolutionary dynamics in XCS



11 multiplexer

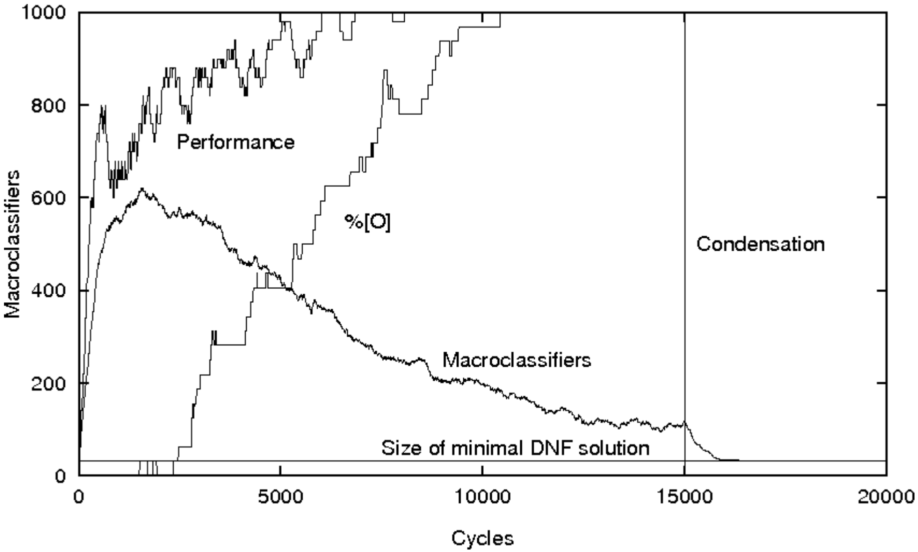
Observations on evolutionary dynamics of XCS

- XCS continues to refine solution after 100% performance reached
- Finds minimal representation
    - but continued crossover and mutation generate extra transient rules

Condensation

- An evolved population normally contains many redundant and low-fitness rules
- Typically transient, but more generated while GA runs
- We can remove them with condensation:
    - Run the system with crossover and mutation turned off [313, 159]
    - I.e. only clone and delete existing rules
    - Next slide shows XCS on 11 multiplexer with condensation
- Other compaction methods [160, 319, 87]

Condensation



Tuning evolutionary search

Manual tuning

- Class imbalances
    - XCS is robust to class imbalances [230]
    - But for high imbalances tuning the GA based on a facetwise model improved performance [230, 232]

Self-tuning Evolutionary Search

- Mutation rate can be adapted during evolution e.g. [140, 141, 138, 63]
- [82] dynamically control use of 2 generalisation operators
    - Each has a control bit specifying whether it can be used
    - Control bits evolve with the rest of the genotype

Non-evolutionary rule discovery

- Covering creates a rule to match an unmatched input
  - First suggested in [129]
- Can create ("seed") the initial population [299, 313, 127]
  - Can also supplement the GA throughout evolution [313]
- Variations
  - [162] p. 42 found covering each action set better when applying XCS to sequential tasks
  - Most covering/seeding is performed as needed
    - Instead [200] select inputs at the center of same-class clusters

Non-evolutionary LCS

- Although LCS were conceived as a way of applying GAs to learning problem [135], not all LCS include a GA!
- Various heuristics used to create and refine rules e.g.
  - YACS [107]
  - MACS [106]
- Methods inspired by psychological models of learning
  - ACS [275, 49] and ACS2 [48]
    - ACS also supplemented by a GA [50, 51]
  - AgentP: specialised LCS for maze tasks [332, 331]

LCS: Credit Assignment

Review: credit assignment in LCS

- Pittsburgh:
  - 1 chromosome = 1 solution
  - Credit assignment is easy
    - If solution is good, chromosome is good
    - Chromosomes only compete
- Michigan:
  - 1 solution = many chromosomes
  - Credit assignment is more complex
    - Chromosomes compete, complement and cooperate
    - How did they contribute to solution?
  - Majority of LCS are Michigan
  - Credit assignment difficulties have been the major issue with them
  - Learning a value function for Reinforcement Learning further complicates fitness evaluation
  - Two major approaches:
    - Strength-based and accuracy-based fitness

Strength-based Michigan systems

- Older (pre-1995)
- Fitness is proportional to the magnitude of reward
- Suffer from difficulties with credit assignment [162]
- Analysis of credit assignment is very complex
- Some incorporate accuracy as a component of fitness
    - but it's still proportional to reward

Accuracy-based Michigan systems

- Newer (starting with XCS in 1995 [313, 52])
- Majority of current research uses them
- Avoid many problems with credit assignment
- Fitness is proportional to the accuracy of reward prediction
- Accuracy estimated from variance in reward
- Overgeneral rules have high variance hence low fitness
- Major limitation: accuracy estimate conflates several things:
    - Overgenerality
    - Noise in the training data
    - Stochasticity in transition function in sequential problems
- Strength-based systems may be *less* affected by noise and stochasticity
- See [162] for analysis of strength and accuracy

Strength and accuracy in Reinforcement Learning

Strength:

- A form of direct policy search (see e.g. [242])
  - Searches in space of policies
- Each rule contributes to policy
- Generalisation is over policy

Accuracy:

- Learns Value Function (VF)
  - Searches for good state-action aggregations to represent VF
  - VF then used to generate policy
- Each rule contributes to VF
- Generalisation is aggregation of state-action pairs in VF

Credit assignment algorithms in XCS

- Classifiers update predictions while training
- Updates in basic XCS: [313, 52]
  - Widrow-Hoff update (for non-sequential problems)
  - Q-learning update (for sequential problems)
- Alternative XCS updates:
  - Average rewards [284, 195]
  - Gradient descent [57, 194]
  - Eligibility traces [92]

Other credit assignment algorithms

- Update in basic XCSF: NLMS (linear piecewise) [317, 318]
- Alternative updates compared in XCSF [187]
  - Classical parameter estimation algs: RLS, and Kalman filter
  - Gain adaptation algs: K1, K2, IDBD, and IDD
- Findings:
  - Kalman filter and RLS have significantly better accuracy
  - Kalman filter produces more compact solutions than RLS

Other LCS:

- UCS: essentially a supervised version of XCS [22]
- Simplified LCS [41]

Evolutionary selection of prediction functions

- Similar to representational ecology work that selects condition types [211]
- XCSFHP (XCSF with Heterogeneous Predictors) [188] selects prediction functions
  - Polynomial functions: linear, quadratic and cubic predictions
  - Also constant, linear and NN predictors
- Selects most suitable predictor for regression and sequential tasks
- Performs almost as well as XCSF using best single predictor

Theoretical results

- XCS without generalisation implements tabular Q-learning [184]
- Computational complexity of XCS (PAC learning) [56]
- Analysis of credit assignment and relation to Reinforcement Learning methods [303, 302, 301, 304]
- Existence of strong and fit overgenerals [162]
  - Rules which are overgeneral yet stronger/fitter than not-overgeneral competitors
  - Only possible under specific circumstances
- Characterising problems which are hard for LCS [114, 164, 161, 162, 23, 15]
- Models of evolutionary dynamics [47, 55, 54, 58, 233, 234]
- Reconstruction of LCS from first principles using probabilistic models [94, 93, 95]

Hierarchies and ensembles of LCS

Hierarchical LCS have been studied for some time e.g.

- [17] reviews early work
- [91] and [89, 90, 88] apply hierarchical LCS to robot control
- [18] uses hierarchical XCSs to learn longer sequences of actions
- All could be reformulated as ensembles

Ensembles of LCS

- The ensembles field studies how to combine predictions [175]
- Recent work on ensembles of LCS [80, 39]

An LCS as an ensemble

- Instead of an ensemble of LCS, we can treat 1 LCS as an ensemble
- In Michigan LCS rules often conflict
- In Pitt LCS rulesets conflict
- Various *conflict resolution* methods have been used
  - Typically a majority vote weighted by fitness (e.g. UCS)
- We can treat conflicting rules (or rulesets) as an ensemble
- Connections beginning to appear [165, 35, 93, 95]

LCS: Conclusions

Conclusions

Inherent difficulties:

- Michigan: credit assignment
- Pitt: run-time

Recent research:

- Integration with mainstream Machine Learning and Reinforcement Learning
- Representations
- Credit assignment algorithms

Future directions:

- Exposing more of the system to evolution
- Further integration with ML, RL, ensembles, memetic algorithms, multi-objective optimisation...

Reading

- No general up-to-date introduction to LCS exists
  - For the basic idea see [113] and the introductory parts of [162] or [54]
  - For a review of early LCS see [18]
- Reviews of LCS research [322, 189, 185]
- The LCS bibliography [163]
- Review of state-of-the-art GBML and empirical comparison to non-evolutionary pattern recognition [236]
- Other comparisons with non-evolutionary methods [25, 118, 257, 316, 21, 22]
- Good introduction to representations and operators ([99] ch. 6)

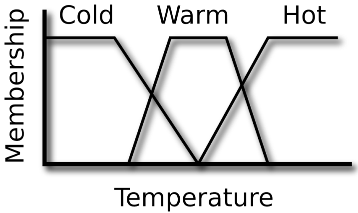
Contents

Genetic Fuzzy Systems

- Fuzzy Logic is a major paradigm in soft computing
- Provides a means of approximate reasoning
- Genetic Fuzzy Systems (GFS) apply evolution to fuzzy systems in various ways
  - GAs, GP and Evolutionary Strategies have all been used
  - We cover genetic Fuzzy Rule-based Systems (FRBS)
    - Also called Learning Fuzzy Classifier Systems (LFCS) [24]
    - Also referred to as e.g. "genetic learning of fuzzy rules" and (for Reinforcement Learning) "fuzzy Q-learning"
    - Like other LCS they evolve if-then rules, but rules are fuzzy
    - An active area, somewhat disjoint from LCS literature
    - Pitt systems common but see e.g. [295, 296, 108, 24, 231, 67, 235]
  - We briefly cover genetic fuzzy NNs
  - We won't cover genetic fuzzy clustering [77]

Fuzzy sets

- Ordinary scalar values are called *crisp* values.
- A membership function defines the degree of match between crisp values and a set of fuzzy linguistic *terms*
- The set of terms is a *fuzzy set*



- Each crisp value matches *each* term to some degree [0,1]
- Fuzzification: computing the membership of each term
    - Can be considered a form of discretisation
- Defuzzification: computing a crisp value from fuzzy sets

Fuzzy rules

Condition/action (IF-THEN) rules composed of:

- A set of linguistic variables (e.g. temperature, humidity)
- Which can each take on linguistic terms (e.g. cold, warm, hot)

Examples:

- IF temperature IS cold AND humidity IS high THEN heater IS high
- IF temperature IS warm AND humidity IS low THEN heater IS medium

Types of rules:

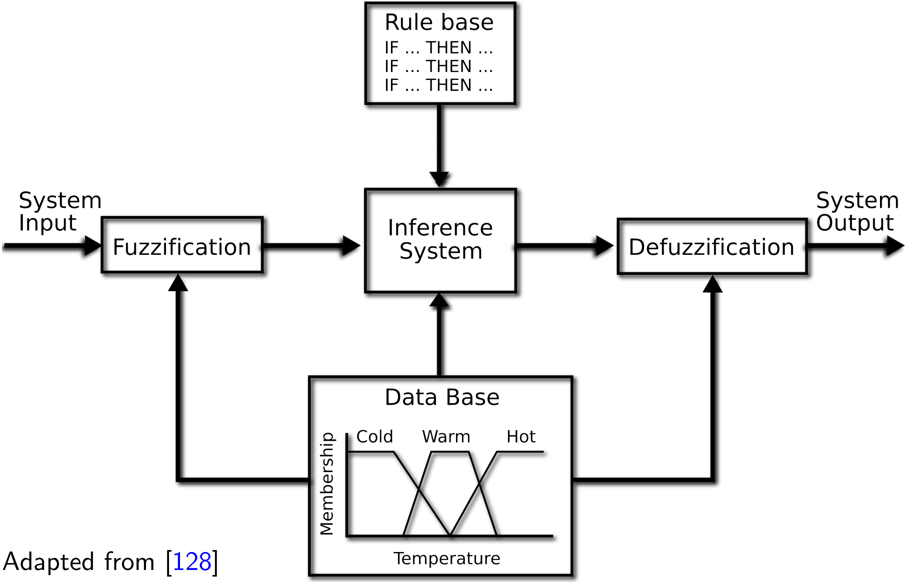
- Linguistic or Mamdani: condition and actions contain fuzzy terms (as above)
- Takagi-Sugeno: condition contains fuzzy terms, action is a function of condition variables
- Approximate or Scatter Partition: instead of linguistic terms condition and action use fuzzy sets directly

Fuzzy Rule-Based System

An FRBS consists of:

- A Rule Base (RB) of fuzzy rules
- A Data Base (DB) of linguistic terms and their membership functions
- Together the RB and DB are the *knowledge base* (KB)
- A fuzzy inference system which maps from fuzzy inputs to a fuzzy output

Fuzzy Rule-Based System



Adapted from [128]

Evolution of FRBS

We distinguish:

1. Genetic tuning
2. Genetic learning of DB, RB or inference engine parameters

See [99] or [128] for further details

Genetic tuning

Concept:

- First train a hand-crafted FRBS
- Then evolve the DB (linguistic terms and membership functions) to improve performance
- Do not alter the rule base

Approaches:

- Adjust the shape of the membership functions
- Adjust parameterised expressions in the (adaptive) inference system
- Adapt defuzzification methods

Genetic learning

Concept:

- Evolve DB, RB or inference engine parameters

Approaches:

- Genetic rule learning
  - Usually predefine the DB by hand and evolve the RB
- Genetic rule selection
  - Use the GA to remove irrelevant, redundant, incorrect or conflicting rules
  - Similar role to condensation in LCS
- Genetic KB learning
  - Learn both the DB and RB. Either:
    - learn the DB first, then learn the RB or
    - iteratively learn DBs and evaluate each one by learning an RB using it

Simultaneous genetic learning

- Simultaneous KB learning
  - Learn the DB and RB simultaneously [221]
- Simultaneous genetic learning of KB components and inference engine parameters [136]
- Simultaneous learning may get better results but the larger search space makes it slow and difficult

Fuzzy fitness

[254] claims:

- Existing GFS are applied to crisp data
  - The benefits of GFS here are only linguistic interpretability
- But GFS can outperform other methods on fuzzy data
  - GFS should use fuzzy fitness functions in such cases
  - They propose this as a new class of GFS to add to the taxonomy of [128]
- They identify 3 cases:
  1. "crisp data with hand-added fuzziness
  2. transformations of data based on semantic interpretations of fuzzy sets
  3. inherently fuzzy data" p. 558

Genetic Neuro-Fuzzy Systems

- Neuro-Fuzzy System (NFS): any combination of fuzzy logic and neural networks
- Also called Fuzzy Neural Networks (FNNs)

Example Genetic NFS:

- See [77] for an introduction
- [198] uses a GA to minimise the error in a FNN
- [121] uses both a GA and backprop to minimise error
- [241] optimises a fuzzy expert system using a GA and NN
- [221] uses NN to approximate fitness function for GA which adapts membership functions and control rules
- [199] reviews the three areas from the perspective of intelligent control
- [125] discusses the combination of the three
- [158] introduces Fuzzy All-permutations Rule-Bases (FARBs); mathematically equivalent to NNs

Active areas within GFS

Herrera [128] p. 38 lists:

1. "Multiobjective genetic learning of FRBSs: interpretability-precision trade-off
2. GA-based techniques for mining fuzzy association rules and novel data mining approaches
3. Learning genetic models based on low quality data (e.g. noisy data)
4. Genetic learning of fuzzy partitions and context adaptation
5. Genetic adaptation of inference engine components
6. Revisiting the Michigan-style GFSs"

Current issues for GFS

Herrera [128] p. 42 lists:

1. Human readability
2. New data mining tasks: frequent and interesting pattern mining, mining data streams ...
3. Dealing with high dimensional data

Reading 1

- Seminal papers from 1991: [128]
  - Genetic tuning of the DB [154]
  - Michigan [295]
  - Pittsburgh [287]
  - Relational matrix-based FRBS [243]
- Geyer-Schulz's 1997 book on Michigan fuzzy LCS learning RBs with GP [108]
- Bonarini's 2000 introduction from an LCS perspective [24]
- Mitra and Hayashi's 2000 survey of neuro-fuzzy rule generation methods [220]

Reading 2

- Cordon et al.'s 2001 book on Genetic Fuzzy Systems in general [77]
- Angelov's 2002 book on evolving FRBS [7]
- Ch. 10 of Freitas' 2002 book on evolutionary data mining [99]
- Herrera's 2008 survey article on GFS [128]
  - Lists more key reading
- Kolman and Margaliot's 2009 book on the neuro-fuzzy FARB approach [158]

Contents

Conclusions

- GBML is very diverse and active
- Constituent areas of GBML should interact more
- Much integration with Machine Learning & Artificial Intelligence has taken place in the last 10 years
  - More is needed
- Integration with ensembles is natural but only just beginning
- Use of multi-objective EAs spreading but more needed

Difficulties for GBML

- Speed of learning
    - EAs are much slower than most methods
    - Sometimes this matters little (off-line learning)
    - Sometimes it's critical (stream mining)
    - Various methods to speed them up exist (see e.g. [99] §12.1.3)
- Theory
    - EA theory is notoriously difficult
    - When coupled with other processes things are even more complex

Research directions 1

- Speed
- Theory
- Meta-learning / hyper-heuristics e.g.
  - Evolution of bias (e.g. selection of representation)
  - Evolving problem class specific heuristics and learning rules
  - Other forms of self-adaptation
- Data preparation
  - Freitas ([99] §12.2.1) argues:
    - attribute construction is a promising area for GBML
    - filter methods for feature selection are faster than wrappers and deserve more GBML research

Research directions 2

- Integration with ensembles, multi-objective optimisation, memetics, meta-learning/hyperheuristics, EDAs, Machine Learning & Artificial Intelligence
- Many specialised learning problems little- or un-explored with GBML e.g.
  - Ranking
  - Semi-supervised learning
  - Transductive learning
  - Inductive transfer
  - Learning to learn
  - Stream mining
  - ...

Contents

Glossary

Chromosome	An individual's genes
EA	Evolutionary Algorithm
EDA	Evolution of Distribution Algorithm
FRBS	Fuzzy Rule-Based System
GA	Genetic Algorithm
GCCL	Genetic Cooperative-Competitive Learning
GBML	Genetics-based Machine Learning
Genotype	An individual's genes
GFS	Genetic Fuzzy System
GP	Genetic Programming
IRL	Iterative Rule Learning
LCS	Learning Classifier System
Michigan approach	Solution is a set of chromosomes
Phenotype	An individual's body
Pittsburgh approach	Solution is a single chromosome
NN	Neural Network
SL	Supervised Learning

Contents

[1] H.A. Abbass.
Speeding up backpropagation using multiobjective evolutionary algorithms.
Neural Computation, 15(11):2705–2726, 2003.
79, 102

[2] D.H. Ackley and M.L. Littman.
Interactions between learning and evolution.
In C. Langton, C. Taylor, S. Rasmussen, and J. Farmer, editors, *Artificial Life II: Santa Fe Institute Studies in the Sciences of Complexity*, volume 10, pages 487–509. Addison Wesley, 1992.
41

[3] J. Aguilar-Ruiz, J. Riquelme, and M. Toro.
Evolutionary learning of hierarchical decision rules.
IEEE Transactions on Systems, Man and Cybernetics, Part B, 33(2):324–331, 2003.
111, 117

[4] Manu Ahluwalia and Larry Bull.
A Genetic Programming-based Classifier System.
In Banzhaf et al. [16], pages 11–18.
121

[5] H.C. Andersen and A.C. Tsoi.
A constructive algorithm for the training of a multi-layer perceptron based on the genetic algorithm.
Complex Systems, 7(4):249–268, 1993.
88

[6] P.J. Angeline, G.M. Sauders, and J.B. Pollack.
An evolutionary algorithm that constructs recurrent neural networks.
IEEE Trans. Neural Networks, 5:54–65, 1994.
93

[7] Plamen Angelov.
Evolving Rule-based Models. A tool for design of flexible adaptive systems, volume 92 of *Studies in fuzziness and soft computing*.
Springer-Verlag, 2002.
182

[8] A. Asuncion and D.J. Newman.
UCI machine learning repository http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2009.
80

[9] J. Bacardit, E.K. Burke, and N. Krasnogor.
Improving the scalability of rule-based evolutionary learning.
Memetic Computing, 1(1):55–67, 2009.
33, 139

[10] J. Bacardit, M. Stout, J.D. Hirst, and N. Krasnogor.
Data mining in proteomics with learning classifier systems.
In L. Bull, E. Bernadó Mansilla, and J. Holmes, editors, *Learning Classifier Systems in Data Mining*, pages 17–46. Springer, 2008.
139

[11] J. Bacardit, M. Stout, J.D. Hirst, A. Valencia, R.E. Smith, and N. Krasnogor.
Automated alphabet reduction for protein datasets.
BMC Bioinformatics, 10(6), 2009.
53

[12] Jaume Bacardit.
Pittsburgh Genetic-Based Machine Learning in the Data Mining era: Representations, generalization, and run-time.
PhD thesis, Universitat Ramon Llull, 2004.
117, 119, 121, 122, 124, 139

[13] Jaume Bacardit, David E. Goldberg, and Martin V. Butz.
Improving the performance of a pittsburgh learning classifier system using a default rule.
In Tim Kovacs, Xavier LLòra, Keiki Takadama, Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Learning Classifier Systems. International Workshops, IWLCS 2003-2005, Revised Selected Papers*, volume 4399 of *LNCS*, pages 291–307. Springer, 2007.
119, 124

[14] Jaume Bacardit and Natalio Krasnogor.
Empirical evaluation of ensemble techniques for a pittsburgh learning classifier system.
In Jaume Bacardit, Ester Bernadó-Mansilla, Martin Butz, Tim Kovacs, Xavier Llorà, and Keiki Takadama, editors, *Learning Classifier Systems. 10th and 11th International Workshops (2006-2007)*, volume 4998/2008 of *Lecture Notes in Computer Science*, pages 255–268. Springer, 2008.
139

[15] A.J. Bagnall and Z.V. Zatuchna.
On the classification of maze problems.
In L. Bull and T. Kovacs, editors, *Applications of Learning Classifier Systems*, pages 307–316. Springer, 2005.
161

[16] W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors.
GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann, 1999.
191, 219

[17] Alwyn Barry.
Hierarchy Formulation Within Classifiers System – A Review.
In E. G. Goodman, V. L. Uskov, and W. F. Punch, editors, *Proceedings of the First International Conference on Evolutionary Algorithms and their Application EVCA'96*, pages 195–211, Moscow, 1996. The Presidium of the Russian Academy of Sciences.
162

[18] Alwyn Barry.
XCS Performance and Population Structure within Multiple-Step Environments.
PhD thesis, Queens University Belfast, 2000.
162, 166

[19] Thomas Beielstein and Shandor Markon.
Threshold selection, hypothesis tests and DOE methods.
In *2002 Congress on Evolutionary Computation*, pages 777–782, 2002.
47

[20] R.K. Belew, J. McInerney, and N.N. Schraudolph.
Evolving networks: using the genetic algorithm with connectionistic learning.
In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Proceedings of the 2nd Conference on Artificial Life*, pages 51–548. Addison-Wesley, 1992.
95

[21] Ester Bernadó, Xavier Llorà, and Josep M. Garrell.
XCS and GALE: A Comparative Study of Two Learning Classifier Systems on Data Mining.
In Lanzi et al. [192], pages 115–132.
166

[22] Ester Bernadó-Mansilla and Josep M. Garrell-Guiu.
Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks.
Evolutionary Computation, 11(3):209–238, 2003.
159, 166

[23] Ester Bernadó-Mansilla and T.K. Ho.
Domain of competence of XCS classifier system in complexity measurement space.
IEEE Trans. Evolutionary Computation, 9(1):82–104, 2005.
161

[24] Andrea Bonarini.
An Introduction to Learning Fuzzy Classifier Systems.
In Lanzi et al. [190], pages 83–104.
122, 168, 181

[25] Pierre Bonelli and Alexandre Parodi.
An Efficient Classifier System and its Experimental Comparison with two Representative learning methods on three medical domains.
In Booker and Belew [30], pages 288–295.
166

[26] Lashon B. Booker.
Triggered rule discovery in classifier systems.
In Schaffer [258], pages 265–274.
141

[27] Lashon B. Booker.
Representing Attribute-Based Concepts in a Classifier System.
In Gregory J. E. Rawlins, editor, *Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA91)*, pages 115–127. Morgan Kaufmann: San Mateo, 1991.
116

[28] Lashon B. Booker.
Adaptive value function approximations in classifier systems.
In *GECCO '05: Proceedings of the 2005 workshops on Genetic and evolutionary computation*, pages 90–91. ACM, 2005.
121

[29] Lashon B. Booker.
Approximating value functions in classifier systems.
In L. Bull and T. Kovacs, editors, *Foundations of Learning Classifier Systems*, volume 183/2005 of *Studies in Fuzziness and Soft Computing*, pages 45–61. Springer, 2005.
121

[30] Lashon B. Booker and Richard K. Belew, editors.
Proceedings of the 4th International Conference on Genetic Algorithms (ICGA91). Morgan Kaufmann, July 1991.
194, 209, 237

[31] M.C.J. Bot and W.B. Langdon.
Application of genetic programming to induction of linear classification trees.
In *Genetic Programming: Proceedings of the 3rd European Conference (EuroGP 2000)*, volume 1802 of *LNCS*, pages 247–258. Springer, 2000.
64

[32] L. Breiman.
Bagging predictors.
Machine Learning, 24(2):123–140, 1996.
77

[33] L. Breiman.
Arcing classifiers.
Annals of Statistics, 26(3):801–845, 1998.
77

[34] Gavin Brown.
Ensemble learning.
In Claude Sammut and Geoffrey Webb, editors, *Encyclopedia of Machine Learning*. Springer-Verlag, 2010.
83

[35] Gavin Brown, Tim Kovacs, and James Marshall.
UCSpv: Principled Voting in UCS Rule Populations.
In Hod Lipson et al., editor, *GECCO'07: the Genetic and Evolutionary Computation Conference*, pages 1774–1781. ACM, 2007.
163

[36] Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao.
Diversity creation methods: A survey and categorisation.
Journal of Information Fusion (Special issue on Diversity in Multiple Classifier Systems), 6(1):5–20, 2005.
73, 83

[37] L. Bull.
On dynamical genetic programming: Simple boolean networks in learning classifier systems.
International Journal of Parallel, Emergent and Distributed Systems, 24(5):421–442, 2009.
121

[38] L. Bull, P.L. Lanzi, and T. O'Hara.
Anticipation mappings for learning classifier systems.
In *Proceedings of the 2007 congress on evolutionary computation (CEC2007)*, pages 2133–214. IEEE, 2007.
129

[39] L. Bull, M. Studley, T. Bagnall, and I. Whittley.
On the use of rule-sharing in learning classifier system ensembles.
IEEE Trans. Evolutionary Computation, 11:496–502, 2007.
162

[40] Larry Bull.
Lookahead And Latent Learning In ZCS.
In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 897–904, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
129

[41] Larry Bull.
Two Simple Learning Classifier Systems.
In Larry Bull and Tim Kovacs, editors, *Foundations of Learning Classifier Systems*, number 183 in Studies in Fuzziness and Soft Computing, pages 63–90. Springer-Verlag, 2005.
159

[42] Larry Bull.
On lookahead and latent learning in simple lcs.
In Jaume Bacardit, Ester Bernadó-Mansilla, Martin Butz, Tim Kovacs, Xavier Llorà, and Keiki Takadama, editors, *Learning Classifier Systems. 10th and 11th International Workshops (2006-2007)*, volume 4998/2008 of *Lecture Notes in Computer Science*, pages 154–168. Springer, 2008.
129

[43] Larry Bull and Toby O'Hara.
Accuracy-based neuro and neuro-fuzzy classifier systems.
In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 905–911. Morgan Kaufmann Publishers, 9-13 July 2002.
122

[44] Edmund K. Burke, Mathew R. Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and John R. Woodward.
Exploring hyper-heuristic methodologies with genetic programming.
In C. Mumford and L. Jain, editors, *Collaborative Computational Intelligence*. Springer, 2009.
68

[45] E.K. Burke and G. Kendall.
Introduction.
In E.K. Burke and G. Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 5–18. Springer, 2005.
29

[46] E.K. Burke, G. Kendall, J. Newall, E. Hart, P. Russ, and S. Schulenburg.
Hyper-heuristics: An emerging direction in modern search technology.
In F. Glover and G. Kochenberger, editors, *Handbook of Meta-heuristics*, pages 457–474. Kluwer, 2003.
29

[47] Martin Butz, Tim Kovacs, Pier Luca Lanzi, and Stewart W. Wilson.
Toward a theory of generalization and learning in XCS.
IEEE Transactions on Evolutionary Computation, 8(1):8–46, 2004.
161

[48] Martin V. Butz.
An Algorithmic Description of ACS2.
In Lanzi et al. [192], pages 211–229.
152

[49] Martin V. Butz.
Anticipatory learning classifier systems.
Kluwer Academic Publishers, 2002.
129, 152

[50] Martin V. Butz, David E. Goldberg, and Wolfgang Stolzmann.
Introducing a Genetic Generalization Pressure to the Anticipatory Classifier System – Part 1: Theoretical Approach.
In Whitley et al. [307], pages 34–41.
Also Technical Report 2000005 of the Illinois Genetic Algorithms Laboratory.
152

[51] Martin V. Butz, David E. Goldberg, and Wolfgang Stolzmann.
Introducing a Genetic Generalization Pressure to the Anticipatory Classifier System – Part 2: Performance Analysis.
In Whitley et al. [307], pages 42–49.
Also Technical Report 2000006 of the Illinois Genetic Algorithms Laboratory.
152

[52] Martin V. Butz and Stewart W. Wilson.
An Algorithmic Description of XCS.
In Lanzi et al. [191], pages 253–272.
143, 156, 158

[53] M.V. Butz.
Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system.
In H.G. Beyer et al., editor, *Proc. genetic and evolutionary computation conference (GECCO 2005)*, pages 1835–1842. ACM, 2005.
121

[54] M.V. Butz.
Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design.
Studies in Fuzziness and Soft Computing. Springer–Verlag, 2006.
161, 166

[55] M.V. Butz, D.E. Goldberg, and P.L. Lanzi.
Bounding learning time in XCS.
In *Genetic and evolutionary computation (GECCO 2004)*, volume 3103/2004 of *LNCS*, pages 739–750. Springer, 2004.
161

[56] M.V. Butz, D.E. Goldberg, and P.L. Lanzi.
Computational complexity of the XCS classifier system.
In Larry Bull and Tim Kovacs, editors, *Foundations of Learning Classifier Systems*, number 183 in Studies in Fuzziness and Soft Computing, pages 91–126. Springer-Verlag, 2005.
161

[57] M.V. Butz, D.E. Goldberg, and P.L. Lanzi.
Gradient descent methods in learning classifier systems: improving XCS performance in multistep problems.
IEEE Trans. Evolutionary Computation, 9(5):452–473, 2005.
158

[58] M.V. Butz, D.E. Goldberg, P.L. Lanzi, and K. Sastry.
Problem solution sustenance in XCS: Markov chain analysis of niche support distributions and the impact on computational complexity.
Genetic Programming and Evolvable Machines, 8(1):5–37, 2007.
161

[59] M.V. Butz, P.L. Lanzi, and S.W. Wilson.
Hyper-ellipsoidal conditions in XCS: rotation, linear approximation, and solution structure.
In M. Cattolico, editor, *Proc. genetic and evolutionary computation conference (GECCO 2006)*, pages 1457–1464. ACM, 2006.
121

[60] M.V. Butz and M. Pelikan.
Studying XCS/BOA learning in boolean functions: structure encoding and random boolean functions.
In M. Cattolico et al., editor, *Genetic and evolutionary computation conference, GECCO 2006*, pages 1449–1456. ACM, 2006.
142

[61] M.V. Butz, M. Pelikan, X. Llorà, and D.E. Goldberg.
Extracted global structure makes local building block processing effective in XCS.
In H.G. Beyer and U.M. O'Reilly, editors, *Genetic and evolutionary computation conference, GECCO 2005*, pages 655–662. ACM, 2005.
142

[62] M.V. Butz, M. Pelikan, X. Llorà, and D.E. Goldberg.
Automated global structure extraction for effective local building block processing in XCS.
Evolutionary Computation, 14(3):345–380, 2006.
142

[63] M.V. Butz, P. Stalph, and P.L. Lanzi.
Self-adaptive mutation in XCSF.
In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1365–1372. ACM, 2008.
150

[64] E. Cantu-Paz and C. Kamath.
Inducing oblique decision trees with evolutionary algorithms.
IEEE Transactions on Evolutionary Computation, 7(1):54–68, 2003.
18, 55

[65] Erick Cantú-Paz.
Feature subset selection by estimation of distribution algorithms.
In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 303–310. Morgan Kaufmann, 2002.
53

[66] Rich Caruana and Alexandru Niculescu-Mizil.
An empirical comparison of supervised learning algorithms.
In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.
74

[67] J. Casillas, B. Carse, and L. Bull.
Fuzzy-XCS: a michigan genetic fuzzy system.
IEEE Trans. Fuzzy Systems, 15:536–550, 2007.
168

[68] P.A. Castilloa, J.J. Merelo, M.G. Arenas, and G. Romero.
Comparing evolutionary hybrid systems for design and optimization of multilayer perceptron structure along training parameters.
Information Sciences, 177(14):2884–2905, 2007.
97

[69] D. Chalmers.
The evolution of learning: An experiment in genetic connectionism.
In E. Touretsky, editor, *Proc. 1990 Connectionist Models Summer School*, pages 81–90. Morgan Kaufmann, 1990.
98

[70] Arjun Chandra and Xin Yao.
Ensemble learning using multi-objective evolutionary algorithms.
Journal of Mathematical Modelling and Algorithms, 5(4):417–445, 2006.
Introduces DIVACE.
79, 83, 102

[71] Arjun Chandra and Xin Yao.
Evolving hybrid ensembles of learning machines for better generalisation.
Neurocomputing, 69(7–9):686–700, 2006.
Introduces DIVACE-II.
73, 79, 82, 103

[72] S. Cho and K. Cha.
Evolution of neural net training set through addition of virtual samples.
In *Proc. 1996 IEEE Int. Conf. Evol. Comp., ICEC'96*, pages 685–688. IEEE, 1996.
55

[73] S.-B. Cho.
Pattern recognition with neural networks combined by genetic algorithm.
Fuzzy Sets and Systems, 103:339–347, 1999.
See Kuncheva2004a p.167.
78

[74] Sung-Bae Cho and Chanho Park.
Speciated GA for optimal ensemble classifiers in DNA microarray classification.
In *Congress on Evolutionary Computation (CEC 2004)*, volume 1, pages 590–597, 2004.
73

[75] Dave Cliff and Susi Ross.
Adding Temporary Memory to ZCS.
Adaptive Behavior, 3(2):101–150, 1994.
Also technical report: ftp://ftp.cogs.susx.ac.uk/pub/reports/csrp/csrp347.ps.Z.
132

[76] A.L. Corcoran and S. Sen.
Using real-valued genetic algorithms to evolve rule sets for classification.
In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 120–124. IEEE Press, 1994.
117

[77] Oscar Cordón, Francisco Herrera, Frank Hoffmann, and Luis Magdalena.
Genetic Fuzzy Systems.
World Scientific, 2001.
168, 178, 182

[78] Henry Brown Cribbs III and Robert E. Smith.
Classifier system renaissance: New analogies, new directions.
In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 547–552, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
122

[79] Hai Huong Dam, Hussein A. Abbass, Chris Lokan, and Xin Yao.
Neural-based learning classifier systems.
IEEE Trans. Knowl. Data Eng., 20(1):26–39, 2008.
122

[80] H.H. Dam, H.A. Abbass, and C. Lokan.
DXCS: an XCS system for distributed data mining.
In H.G. Beyer and U.M. O'Reilly, editors, *Genetic and evolutionary computation conference, GECCO 2005*, pages 1883–1890, 2005.
162

[81] A. Dasdan and K. Oflazer.
Genetic synthesis of unsupervised learning algorithms.
Technical Report BU-CEIS-9306, Department of Computer Engineering and Information Science, Bilkent University, Ankara, 1993.
98

[82] Kenneth A. De Jong, William M. Spears, and Dianna F. Gordon.
Using Genetic Algorithms for Concept Learning.
Machine Learning, 3:161–188, 13.
150

[83] T.G. Dietterich.
Machine-learning research: four current directions.
AI Magazine, 18(4):97–136, 1998.
73, 76

[84] F. Divina, M. Keijzer, and E. Marchiori.
Non-universal suffrage selection operators favor population diversity in genetic algorithms.
In *Benelearn 2002: Proceedings of the 12th Belgian-Dutch Conference on Machine Learning (Technical report UU-CS-2002-046)*, pages 23–30, 2002.
55

[85] F. Divina, M. Keijzer, and E. Marchiori.
A method for handling numerical attributes in GA-based inductive concept learners.
In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, pages 898–908. Springer-Verlag, 2003.
55, 117

[86] Federico Divina and Elena Marchiori.
Evolutionary concept learning.
In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 343–350, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
55

[87] P.W. Dixon, D. Corne, and M.J. Oates.
A ruleset reduction algorithm for the XCS learning classifier system.
In P.L. Lanzi, W. Stolzmann, and S.W. Wilson, editors, *Learning classifier systems, 5th international workshop (IWLCS 2002)*, volume 2661 of *LNCS*, pages 20–29. Springer, 2002.
148

[88] Jean-Yves Donnart.
Cognitive Architecture and Adaptive Properties of an Motivationally Autonomous Animat.
PhD thesis, Université Pierre et Marie Curie. Paris, France, 1998.
162

[89] Jean-Yves Donnart and Jean-Arcady Meyer.
Hierarchical-map Building and Self-positioning with MonaLysa.
Adaptive Behavior, 5(1):29–74, 1996.
162

[90] Jean-Yves Donnart and Jean-Arcady Meyer.
Learning Reactive and Planning Rules in a Motivationally Autonomous Animat.
IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics, 26(3):381–395, 1996.
162

[91] Marco Dorigo and Marco Colombetti.
Robot Shaping: An Experiment in Behavior Engineering.
MIT Press/Bradford Books, 1998.
132, 162

[92] J. Drugowitsch and A. Barry.
XCS with eligibility traces.
In H.G. Beyer and U.M. O'Reilly, editors, *Genetic and evolutionary computation conference, GECCO 2005*, pages 1851–1858. ACM, 2005.
158

[93] Jan Drugowitsch.
Design and Analysis of Learning Classifier Systems: A Probabilistic Approach.
Springer, 2008.
161, 163

[94] Jan Drugowitsch and Alwyn Barry.
A Formal Framework and Extensions for Function Approximation in Learning Classifier Systems.
Machine Learning, 70(1):45–88, 2007.
161

[95] Narayanan E. Edakunni, Tim Kovacs, Gavin Brown, and James A.R. Marshall.
Modeling UCS as a mixture of experts.
In *Proceedings of the 2009 Genetic and Evolutionary Computation Conference (GECCO'09)*, pages 1187–1194. ACM, 2009.
161, 163

[96] Dario Floreano, Peter Dürr, and Claudio Mattiussi.
Neuroevolution: from architectures to learning.
Evolutionary Intelligence, 1(1):47–62, 2008.
40, 87, 91, 92, 95, 106, 107

[97] T.C. Fogarty.
An incremental genetic algorithm for real-time learning.
In *Proc. Sixth Int. Workshop on Machine Learning*, pages 416–419, 1989.
48

[98] G. Folino, C. Pizzuti, and G. Spezzano.
Ensemble techniques for parallel genetic programming based classifiers.
In *Proc. European Conf. on Genetic Programming (EuroGP'03)*, pages 59–69, 2003.
66

[99] A.A. Freitas.
Data Mining and Knowledge Discovery with Evolutionary Algorithms.
Spinger-Verlag, Berlin, 2002.
16, 17, 20, 53, 58, 68, 71, 166, 173, 182, 185, 186

[100] A.A. Freitas.
A survey of evolutionary algorithms for data mining and knowledge discovery.
In A. Ghosh and S. Tsutsui, editors, *Advances in Evolutionary Computation*, pages 819–845. Springer-Verlag, 2002.
34, 53

[101] Y. Freund and R. Schapire.
Experiments with a new boosting algorithm.
In *Proc. of the INt. Conf. on Machine Learning (ICML'96)*, pages 148–156, 1996.
77

[102] Y. Freund and R. Schapire.
A short introduction to boosting.
Journal of the Japanese Society for Artificial Intelligence, 14(5):771–780, 1999.
77

[103] J. Fürnkranz.
Integrative windowing.
Journal of Artificial Intelligence Research, 8:129–164, 1998.
139

[104] Christian Gagné, Michèle Sebag, Marc Schoenauer, and Marco Tomassini.
Ensemble learning for free with evolutionary algorithms?
In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1782–1789. ACM, 2007.
79, 80, 81

[105] C. Gathercole and P. Ross.
Tackling the boolean even n parity problem with genetic programming and limited-error fitness.
In J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, and R.L. Riolo, editors, *Genetic Programming 1997: Proc. Second Annual Conference*, pages 119–127. Morgan Kaufmann, 1997.
67

[106] Pierre Gérard and Olivier Sigaud.
Designing efficient exploration with MACS: Modules and function approximation.
In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS*, pages 1882–1893. Springer-Verlag, 2003.
129, 152

[107] Pierre Gerard, Wolfgang Stolzmann, and Olivier Sigaud.
YACS, a new learning classifier system using anticipation.
Journal of Soft Computing, 6(3–4):216–228, 2002.
129, 152

[108] Andreas Geyer-Schulz.
Fuzzy Rule-Based Expert Systems and Genetic Machine Learning.
Physica Verlag, 1997.
168, 181

[109] Attilio Giordana and Filippo Neri.
Search-Intensive Concept Induction.
Evolutionary Computation, 3:375–416, 1995.
38

[110] Attilio Giordana and L. Saitta.
Learning disjunctive concepts by means of genetic algorithms.
In *Proc. Int. Conf. on Machine Learning*, pages 96–104, 1994.
141

[111] R. Giraldez, J. Aguilar-Ruiz, and J. Riquelme.
Natural coding: A more efficient representation for evolutionary learning.
In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, pages 979–990.
Springer-Verlag, 2003.
117

[112] C. Giraud-Carrier and J. Keller.
Meta-learning.
In J. Meij, editor, *Dealing with the data flood*. STT/Beweton, 2002.
29

[113] David E. Goldberg.
Genetic Algorithms in Search, Optimization, and Machine Learning.
Addison-Wesley, Reading, Mass., 1989.
20, 166

[114] David E. Goldberg, Jeffrey Horn, and Kalyanmoy Deb.
What Makes a Problem Hard for a Classifier System?
In *Collected Abstracts for the First International Workshop on Learning Classifier System (IWLCS-92)*, 1992.
(Also technical report 92007 Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign).
Available from ENCORE (ftp://ftp.krl.caltech.edu/pub/EC/Welcome.html) in the section on Classifier Systems.
161

[115] David Perry Greene and Stephen F. Smith.
Competition-based induction of decision models from examples.
Machine Learning, 13:229–257, 1993.
34, 38, 110, 139

[116] David Perry Greene and Stephen F. Smith.
Using Coverage as a Model Building Constraint in Learning Classifier Systems.
Evolutionary Computation, 2(1):67–91, 1994.
38

[117] D.P. Greene and S.F. Smith.
A genetic system for learning models of consumer choice.
In *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, pages 217–223.
Morgan Kaufmann, 1987.
139

[118] A. Greenyer.
The use of a learning classifier system JXCS.
In P. van der Putten and M. van Someren, editors, *CoIL Challenge 2000: The Insurance Company Case*. Leiden Institute of Advanced Computer Science, June 2000.
Technical report 2000-09.
166

[119] John J. Grefenstette.
Lamarckian Learning in Multi-Agent Environments.
In Booker and Belew [30], pages 303–310.
http://www.ib3.gmu.edu/gref/publications.html.
41

[120] F. Gruau.
Automatic definition of modular neural networks.
Adaptive Behavior, 3(2):151–183, 1995.
87

[121] D. Hanebeck and K. Schmidt.
Genetic optimization of fuzzy networks.
Fuzzy sets and systems, 79:59–68, 1996.
178

[122] L.K. Hansen and P. Salamon.
Neural network ensembles.
IEEE Trans. Pattern Analysis and Machine Intelligence, pages 993–1001, 1990.
76

[123] W.E. Hart, N. Krasnogor, and J.E. Smith (editors).
Special issue on memetic algorithms.
Evolutionary Computation, 12(3), 2004.
40

[124] William E. Hart, N. Krasnogor, and J.E. Smith, editors.
Recent Advances in Memetic Algorithms, volume 166 of *Studies in Fuzziness and Soft Computing*.
Springer, 2005.
40

[125] Lin He, Ke jun Wang, Hong zhang Jin, Guo bin Li, and X.Z. Gao.
The combination and prospects of neural networks, fuzzy logic and genetic algorithms.
In *IEEE Midnight-Sun Workshop on Soft Computing Methods in Industrial Applications*, pages 52–57. IEEE, 1999.
178

[126] Jörg Heitkötter and David Beasley.
The Hitch-Hiker's Guide to Evolutionary Computation (FAQ for comp.ai.genetic). Accessed 28/2/09.
http://www.aip.de/~ast/EvolCompFAQ/, 2001.
20, 110

[127] J. Hekanaho.
Symbiosis in multimodal concept learning.
In *Proc. 1995 Int. Conf. on Machine Learning (ML'95)*, pages 278–285, 1995.
151

[128] Francisco Herrera.
Genetic fuzzy systems: taxonomy, current research trends and prospects.
Evolutionary Intelligence, 1(1):27–46, 2008.
122, 172, 173, 177, 179, 180, 181, 182

[129] John H. Holland.
Adaptation.
In R. Rosen and F. M. Snell, editors, *Progress in Theoretical Biology*. New York: Plenum, 1976.
151

[130] John H. Holland.
Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems.
In T. Mitchell, R. Michalski, and J. Carbonell, editors, *Machine learning, an artificial intelligence approach. Volume II*, chapter 20, pages 593–623. Morgan Kaufmann, 1986.
69

[131] John H. Holland.
Concerning the emergence of tag-mediated lookahead in classifier systems.
Physica D, 42:188–201, 1990.
129

[132] John H. Holland, Lashon B. Booker, Marco Colombetti, Marco Dorigo, David E. Goldberg, Stephanie Forrest, Rick L. Riolo, Robert E. Smith, Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson.
What is a Learning Classifier System?
In Lanzi et al. [190], pages 3–32.
110

[133] John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and P. R. Thagard.
Induction: Processes of Inference, Learning, and Discovery.
MIT Press, Cambridge, 1986.
118

[134] John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard.
Induction. Processes of Inference, Learning and Discovery.
The MIT Press, 1986.
132

[135] John H. Holland and J. S. Reitman.
Cognitive systems based on adaptive algorithms.
In D. A. Waterman and F. Hayes-Roth, editors, *Pattern-directed Inference Systems*. New York: Academic Press, 1978. Reprinted in: Evolutionary Computation. The Fossil Record. David B. Fogel (Ed.) IEEE Press, 1998. ISBN: 0-7803-3481-7.
152

[136] A. Homaifar and E. Mccormick.
Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms.
IEEE Trans. Fuzzy. Syst., 3(2):129–139, 1995.
176

[137] D. Howard and L. Bull.
On the effects of node duplication and connection-orientated constructivism in neural XCSF.
In M. Keijzer et al., editor, *GECCO-2008: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1977–1984. ACM, 2008.
122

[138] D. Howard, L. Bull, and P.L. Lanzi.
Self-Adaptive Constructivism in Neural XCS and XCSF.
In M. Keijzer et al., editor, *GECCO-2008: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1389–1396. ACM, 2008.
122, 150

[139] Y.-J. Hu.
A genetic programming approach to constructive induction.
In *Genetic Programming 1998: Proceedings of the 3rd Annual Conference*, pages 146–151. Morgan Kaufmann, 1998.
54

[140] J. Hurst and L. Bull.
Self-adaptation in classifier system controllers.
Artificial Life and Robotics, 5(2):109–119, 2003.
150

[141] J. Hurst and L. Bull.
A self-adaptive neural learning classifier system with constructivism for mobile robot control.
In X. Yao et al., editor, *Parallel problem solving from nature (PPSN VIII)*, volume 3242 of *LNCS*, pages 942–951. Springer, 2004.
150

[142] P. Husbands, I. Harvey, D. Cliff, and G. Miller.
The use of genetic algorithms for the development of sensorimotor control systems.
In P. Gaussier and J.-D. Nicoud, editors, *From perception to action*, pages 110–121. IEEE Press, 1994.
87

[143] H. Iba.
Bagging, boosting and bloating in genetic programming.
In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'99)*, pages 1053–1060, 1999.
66

[144] IEEE.
Proceedings of the 2000 Congress on Evolutionary Computation (CEC00). IEEE Press, 2000.
222, 230

[145] H. Ishibuchi and T. Nakashima.
Multi-objective pattern and feature selection by a genetic algorithm.
In *Proceedings of the 2000 Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 1069–1076. Morgan Kaufmann, 2000.
55

[146] M.M. Islam, X. Yao, and K. Murase.
A constructive algorithm for training cooperative neural network ensembles.
IEEE Transactions on Neural Networks, 14:820–834, 2003.
101

[147] A. Jain and D. Zongker.
Feature selection: evaluation, application and small sample performance.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 19(2):153–158, 1997.
53

[148] C.Z. Janikow.
Indictive learning of decision rules in attribute-based examples: a knowledge-intensive genetic algorithm approach.
PhD thesis, University of North Carolina, 1991.
121

[149] C.Z. Janikow.
A knowledge-intensive genetic algorithm for supervised learning.
Machine Learning, 13:189–228, 1993.
34, 55

[150] Y. Jin and B. Sendhoff.
Reducing fitness evaluations using clustering techniques and neural network ensembles.
In *Genetic and Evolutionary Computation Conference (GECCO–2004)*, volume 3102 of *Lecture Notes in Computer Science*, pages 688–699. Springer, 2004.
101

[151] G. John, R. Kohavi, and K. Phleger.
Irrelevant features and the feature subset problem.
In *Proceedings of the 11th International Conference on Machine Learning*, pages 121–129. Morgan Kaufmann, 1994.
53

[152] Kenneth A. De Jong and William M. Spears.
Learning Concept Classification Rules using Genetic Algorithms.
In *Proceedings of the Twelfth International Conference on Artificial Intelligence IJCAI-91*, volume 2, pages 651–656. Morgan Kaufmann, 1991.
119, 121

[153] J.D. Kelly Jr. and L. Davis.
Hybridizing the genetic algorithm and the k nearest neighbors classification algorithm.
In Lashon B. Booker and Richard K. Belew, editors, *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA91)*, pages 377–383. Morgan Kaufmann, July 1991.
54, 55

[154] C. Karr.
Genetic algorithms for fuzzy controllers.
AI Expert, 6(2):26–33, 1991.
181

[155] N. Kasabov.
Evolving Connectionist Systems: The Knowledge Engineering Approach.
Springer, 2007.
86, 107

[156] M. Keijzer and V. Babovic.
Genetic programming, ensemble methods, and the bias/variance/tradeoff – introductory investigation.
In *Proc. of the European Conf. on Genetic Programming (EuroGP'00)*, pages 76–90, 2000.
66

[157] H. Kitano.
Designing neural networks by genetic algorithms using graph generation system.
Journal of Complex System, 4:461–476, 1990.
87

[158] Eyal Kolman and Michael Margaliot.
Knowledge-Based Neurocomputing: A Fuzzy Logic Approach, volume 234 of *Studies in Fuzziness and Soft Computing*.
Springer, 2009.
178, 182

[159] Tim Kovacs.
Evolving Optimal Populations with XCS Classifier Systems.
Master's thesis, University of Birmingham, Birmingham, UK, 1996.
135, 148

[160] Tim Kovacs.
XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions.
In Roy, Chawdhry, and Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 59–68.
Springer-Verlag, London, 1997.
ftp://ftp.cs.bham.ac.uk/pub/authors/T.Kovacs/index.html.
148

[161] Tim Kovacs.
Strength or Accuracy? Fitness calculation in learning classifier systems.
In Lanzi et al. [190], pages 143–160.
161

[162] Tim Kovacs.
Strength or Accuracy: Credit Assignment in Learning Classifier Systems.
Springer, 2004.
34, 116, 141, 151, 155, 156, 161, 166

[163] Tim Kovacs.
A Learning Classifier Systems Bibliography. Department of Computer Science, University of Bristol, 2009.
http://www.cs.bris.ac.uk/˜kovacs/lcs/search.html.
166

[164] Tim Kovacs and Manfred Kerber.
What makes a problem hard for XCS?
In Lanzi et al. [191], pages 80–99.
161

[165] Tim Kovacs and Manfred Kerber.
High classification accuracy does not imply effective genetic search.
In K. Deb et al., editor, *Proceedings of the 2004 Genetic and Evolutionary Computation Conference (GECCO)*, volume 3102 of *LNCS*, pages 785–796. Springer, 2004.
163

[166] John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors.
Genetic Programming 1998: Proceedings of the Third Annual Conference. Morgan Kaufmann, 1998.
219, 239

[167] J.R. Koza.
Genetic Programming: on the programming of computers by means of natural selection.
MIT Press, 1992.
59

[168] J.R. Koza.
Genetic Programming II.
MIT Press, 1994.
71

[169] N. Krasnogor and J.E. Smith.
A tutorial for competent memetic algorithms: model, taxonomy and design issues.
IEEE Transactions on Evolutionary Computation, 9(5):474–488, 2005.
40

[170] Natalio Krasnogor.
Self-generating metaheuristics in bioinformatics: the protein structure comparison case.
Genetic Programming and Evolvable Machines, 5(2):181–201, 2004.
29

[171] Natalio Krasnogor and S. Gustafson.
A study on the use of self-generation in memetic algorithms.
Natural Computing, 3(1):53–76, 2004.
29

[172] K. Krawiec.
Genetic programming-based construction of features for machine learning and knowledge discovery tasks.
Genetic Programming and Evolvable Machines, 3(4):329–343, 2002.
54

[173] A. Krogh and J. Vedelsby.
Neural network ensembles, cross validation and active learning.
Neural Information Processing Systems, 7:231–238, 1995.
76

[174] M. Kudo and J. Skalansky.
Comparison of algorithms that select features for pattern classifiers.
Pattern Recognition, 33:25–41, 2000.
53

[175] Ludmila I. Kuncheva.
Combining Pattern Classifiers: Methods and Algorithms.
Wiley, 2004.
78, 83, 162

[176] I. Kushchu.
An evaluation of evolutionary generalization in genetic programming.
Artificial Intelligence Review, 18(1):3–14, 2002.
69

[177] L. Lam and C.Y. Suen.
Optimal combination of pattern classifiers.
Pattern Recognition Letters, 16:945–954, 1995.
See Kuncheva2004a p.167.
78

[178] Samuel Landau, Olivier Sigaud, and Marc Schoenauer.
ATNoSFERES revisited.
In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2005*, pages 1867–1874. ACM, 2005.
121, 134

[179] William Langdon, Steven Gustafson, and John Koza.
The genetic programming bibliography http://www.cs.bham.ac.uk/ wbl/biblio/, 2009.
71

[180] Pier Luca Lanzi.
An analysis of the memory mechanism of XCSM.
In Koza et al. [166], pages 643–651.
http://ftp.elet.polimi.it/people/lanzi/gp98.ps.gz.
132

[181] Pier Luca Lanzi.
Extending the Representation of Classifier Conditions Part I: From Binary to Messy Coding.
In Banzhaf et al. [16], pages 337–344.
121, 132

[182] Pier Luca Lanzi.
Extending the Representation of Classifier Conditions Part II: From Messy Coding to S-Expressions.
In Banzhaf et al. [16], pages 345–352.
121

[183] Pier Luca Lanzi.
Mining interesting knowledge from data with the XCS classifier system.
In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 958–965, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.
121

[184] Pier Luca Lanzi.
Learning classifier systems from a reinforcement learning perspective.
Journal of Soft Computing, 6(3–4):162–170, 2002.
161

[185] Pier Luca Lanzi.
Learning classifier systems: then and now.
Evolutionary Intelligence, 1(1):63–82, 2008.
166

[186] Pier Luca Lanzi, Daniele Loiacono, Stewart W. Wilson, and David E. Goldberg.
Classifier prediction based on tile coding.
In *Genetic and Evolutionary Computation – GECCO-2006*, pages 1497–1504. ACM, 2006.
121

[187] Pier Luca Lanzi, Daniele Loiacono, Stewart W. Wilson, and David E. Goldberg.
Prediction Update Algorithms for XCSF: RLS, Kalman Filter and Gain Adaptation.
In *Genetic and Evolutionary Computation – GECCO-2006*, pages 1505–1512. ACM, 2006.
159

[188] Pier Luca Lanzi, Daniele Loiacono, and Matteo Zanini.
Evolving classifiers ensembles with heterogeneous predictors.
In Jaume Bacardit, Ester Bernadó-Mansilla, Martin Butz, Tim Kovacs, Xavier Llorà, and Keiki Takadama, editors, *Learning Classifier Systems. 10th and 11th International Workshops (2006-2007)*, volume 4998/2008 of *Lecture Notes in Computer Science*, pages 218–234. Springer, 2008.
160

[189] Pier Luca Lanzi and Rick L. Riolo.
A Roadmap to the Last Decade of Learning Classifier System Research (from 1989 to 1999).
In Lanzi et al. [190], pages 33–62.
166

[190] Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors.
Learning Classifier Systems. From Foundations to Applications, volume 1813 of *LNAI*.
Springer-Verlag, Berlin, 2000.
194, 211, 216, 220, 231, 234

[191] Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors.
Advances in Learning Classifier Systems, volume 1996 of *LNAI*.
Springer-Verlag, Berlin, 2001.
199, 216

[192] Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors.
Advances in Learning Classifier Systems, volume 2321 of *LNAI*.
Springer-Verlag, Berlin, 2002.
194, 198, 240

[193] Pier Luca Lanzi and Stewart W. Wilson.
Toward Optimal Classifier System Performance in Non-Markov Environments.
Evolutionary Computation, 8(4):393–418, 2000.
132

[194] P.L. Lanzi, M.V. Butz, and D.E. Goldberg.
Empirical analysis of generalization and learning in XCS with gradient descent.
In H. Lipson, editor, *Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings*, volume 2, pages 1814–1821. ACM, 2007.
158

[195] P.L. Lanzi and D. Loiacono.
Standard and averaging reinforcement learning in XCS.
In M. Cattolico, editor, *GECCO 2006: Proceedings of the 8th annual conference on genetic and evolutionary computation*, pages 1480–1496. ACM, 2006.
158

[196] P.L. Lanzi and D. Loiacono.
Classifier systems that compute action mappings.
In H. Lipson, editor, *Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings*, pages 1822–1829. ACM, 2007.
122

[197] P.L. Lanzi and S.W. Wilson.
Using convex hulls to represent classifier conditions.
In M. Cattolico, editor, *Proc. genetic and evolutionary computation conference (GECCO 2006)*, pages 1481–1488. ACM, 2006.
121

[198] Z. Liangjie and L. Yanda.
A new global optimizing algorithm for fuzzy neural networks.
Int. J. Electronics, 80(3):393–403, 1996.
178

[199] D.A. Linkens and H.O. Nyongesa.
Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications.
IEE Proceedings - Control Theory and Applications, 143(4):367–386, 1996.
178

[200] Juliet Juan Liu and James Tin-Yau Kwok.
An extended genetic rule induction algorithm.
In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC00)* [144], pages 458–463.
37, 151

[201] Y. Liu and X. Yao.
Ensemble learning via negative correlation.
Neural Networks, 12:1399–1404, 1999.
79, 100

[202] Y. Liu, X. Yao, and T. Higuchi.
Evolutionary ensembles with negative correlation learning.
IEEE Trans. on Evolutionary Computation, 4(4):380–387, 2000.
79, 82, 101

[203] Xavier Llorà.
Genetic Based Machine Learning using Fine-grained Parallelism for Data Mining.
PhD thesis, Enginyeria i Arquitectura La Salle. Ramon Llull University, 2002.
122, 125

[204] Xavier Llorà and Josep M. Garrell.
Knowledge-Independent Data Mining with Fine-Grained Parallel Evolutionary Algorithms.
In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 461–468. Morgan Kaufmann Publishers, 2001.
122

[205] Xavier Llorà, K. Sastry, and D.E. Goldberg.
Binary rule encoding schemes: a study using the compact classifier system.
In F. Rothlauf, editor, *GECCO '05: Proceedings of the 2005 conference on genetic and evolutionary computation, workshop proceedings*, pages 88–89. ACM Press, 2005.
142

[206] Xavier Llorà, K. Sastry, and D.E. Goldberg.
The compact classifier system: scalability analysis and first results.
In F. Rothlauf, editor, *Proceedings of the IEEE congress on evolutionary computation, CEC 2005*, pages 596–603. IEEE, 2005.
142

[207] Xavier Llorà and Stewart W. Wilson.
Mixed Decision Trees: Minimizing Knowledge Representation Bias in LCS.
In Kalyanmoy Deb et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, volume 3103 of *Lecture Notes in Computer Science*, pages 797–809. Springer, 2004.
122, 125

[208] D. Loiacono, A. Marelli, and P.L. Lanzi.
Support vector regression for classifier prediction.
In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1806–1813. ACM, 2007.
121

[209] Urszula Markowska-Kaczmar and Pawelstrok Wnuk-Lipiński.
Artificial intelligence and soft computing - icaisc 2004.
In L. Rutkowski et. al, editor, *Rule Extraction from Neural Network by Genetic Algorithm with Pareto Optimization*, volume 3070/2004 of *LNCS*, pages 450–455. Springer, 2004.
55

[210] R.E. Marmelstein and G.B. Lamont.
Pattern classification using a hybrid genetic algorithm – decision tree approach.
In *Genetic Programming 1998: Proceedings of the 3rd Annual Conference (GP'98)*, pages 223–231. Morgan Kaufmann, 1998.
65

[211] James A. R. Marshall and Tim Kovacs.
A representational ecology for learning classifier systems.
In Maarten Keijzer et al., editor, *Proceedings of the 2006 Genetic and Evolutionary Computation Conference (GECCO 2006)*, pages 1529–1536. ACM, 2006.
121, 126, 160

[212] M.J. Martin-Bautista and M.-A. Vila.
A survey of genetic feature selection in mining issues.
In *Proceedings of the Congress on Evolutionary Computation (CEC'99*, pages 1314–1321. IEEE, 1999.
53

[213] Andrew McCallum.
Reinforcement Learning with Selective Perception and Hidden State.
PhD thesis, University of Rochester, 1996.
131

[214] Ron Meir and Gunnar Rätsch.
An introduction to boosting and leveraging.
In *Advanced lectures on machine learning*, pages 118–183. Springer-Verlag, 2003.
77

[215] Drew Mellor.
A first order logic classifier system.
In F. Rothlauf, editor, *GECCO '05: Proceedings of the 2005 conference on genetic and evolutionary computation*, pages 1819–1826. ACM Press, 2005.
121

[216] Drew Mellor.
Policy transfer with a relational learning classifier system.
In *GECCO Workshops 2005*, pages 82–84. ACM Press, 2005.
121

[217] Drew Mellor.
A learning classifier system approach to relational reinforcement learning.
In Jaume Bacardit, Ester Bernadó-Mansilla, Martin Butz, Tim Kovacs, Xavier Llorà, and Keiki Takadama, editors, *Learning Classifier Systems. 10th and 11th International Workshops (2006-2007)*, volume 4998/2008 of *Lecture Notes in Computer Science*, pages 169–188. Springer, 2008.
121

[218] R.S. Michalski, I. Mozetic, J. Hong, and N. Lavrac.
The AQ15 inductive learning system: an overview and experiments.
Technical Report UIUCDCS-R-86-1260, University of Illinois, 1986.
121

[219] G.F. Miller, P.M. Todd, and S.U. Hegde.
Designing neural networks using genetic algorithms.
In J.D. Schaffer, editor, *Proc. 3rd Int. Conf. Genetic Algorithms and Their Applications*, pages 379–384. Morgan Kaufmann, 1989.
18, 94

[220] Sushmita Mitra and Yoichi Hayashi.
Neurofuzzy rule generation: Survey in soft computing framework.
IEEE Transactions on Neural Networks, 11(3):748–768, 2000.
181

[221] T. Morimoto, J. Suzuki, and Y. Hashimoto.
Optimization of a fuzzy controller for fruit storage using neural networks and genetic algorithms.
Engineering Applications of Art. Int., 10(5):453–461, 1997.
55, 176, 178

[222] S. Nolfi, O. Miglino, and D. Parisi.
Phenotypic plasticity in evolving neural networks.
In P. Gaussier and J.-D. Nicoud, editors, *From perception to action*, pages 146–157. IEEE Press, 1994.
87

[223] T. O'Hara and L. Bull.
Building anticipations in an accuracy-based learning classifier system by use of an artificial neural network.
In *IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 2046–2052. IEEE, 2005.
129

[224] T. O'Hara and L. Bull.
A memetic accuracy-based neural learning classifier system.
In *Proceedings of the IEEE congress on evolutionary computation (CEC 2005)*, pages 2040–2045. IEEE, 2005.
122

[225] Y.-S. Ong, N. Krasnogor, and H. Ishibuchi (editors).
Special issue on memetic algorithms.
IEEE Transactions on Systems, Man and Cybernetics - Part B, 37(1), 2007.
40

[226] Yew-Soon Ong, Meng-Hiot Lim, Ferrante Neri, and Hisao Ishibuchi.
Special issue on memetic algorithms.
Soft Computing, 13(8-9), 2009.
40

[227] Y.S. Ong, M.H. Lim, N. Zhu, and K.W. Wong.
Classification of adaptive memetic algorithms: A comparative study.
IEEE Transactions on Systems Man and Cybernetics – Part B, 36(1):141–152, 2006.
40

[228] D. Opitz and R. Maclin.
Popular ensemble methods: an empirical study.
J. Artificial Intelligence Research, 11:169–198, 1999.
76

[229] D.W. Opitz and J.W. Shavlik.
Generating Accurate and Diverse Members of a Neural-network Ensemble.
Advances in Neural Information Processing Systems, pages 535–541, 1996.
76, 83

[230] A. Orriols-Puig and E. Bernadó-Mansilla.
Bounding XCS's parameters for unbalanced datasets.
In Maarten Keijzer et al., editor, *Proceedings of the 2006 Genetic and Evolutionary Computation Conference (GECCO 2006)*, pages 1561–1568. ACM, 2006.
150

[231] A. Orriols-Puig, J. Casillas, and E. Bernadò-Mansilla.
Fuzzy-UCS: preliminary results.
In H. Lipson, editor, *Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings*, pages 2871–2874. ACM, 2007.
168

[232] A. Orriols-Puig, D.E. Goldberg, K. Sastry, and E. Bernadó-Mansilla.
Modeling XCS in class imbalances: population size and parameter settings.
In H. Lipson et al., editor, *Genetic and evolutionary computation conference, GECCO 2007*, pages 1838–1845. ACM, 2007.
150

[233] A. Orriols-Puig, D.E. Goldberg, K. Sastry, and E. Bernadó-Mansilla.
Modeling XCS in class imbalances: population size and parameter settings.
In H. Lipson, editor, *Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings*, pages 1838–1845. ACM, 2007.
161

[234] A. Orriols-Puig, K. Sastry, P.L. Lanzi, D.E. Goldberg, and E. Bernadò-Mansilla.
Modeling selection pressure in XCS for proportionate and tournament selection.
In H. Lipson, editor, *Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings*, page 18461853. ACM, 2007.
161

[235] Albert Orriols-Puig, Jorge Casillas, and Ester Bernadó-Mansilla.
Evolving fuzzy rules with ucs: Preliminary results.
In Jaume Bacardit, Ester Bernadó-Mansilla, Martin Butz, Tim Kovacs, Xavier Llorà, and Keiki Takadama, editors, *Learning Classifier Systems. 10th and 11th International Workshops (2006-2007)*, volume 4998/2008 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2008.
168

[236] Albert Orriols-Puig, Jorge Casillas, and Ester Bernadó-Mansilla.
Genetic-based machine learning systems are competitive for pattern recognition.
Evolutionary Intelligence, 1(3):209–232, 2008.
19, 166

[237] S. Pal and D. Bhandari.
Genetic algorithms with fuzzy fitness function for object extraction using cellular networks.
Fuzzy Sets and Systems, 65(2–3):129–139, 1994.
87

[238] Gisele L. Pappa and Alex A. Freitas.
Automating the Design of Data Mining Algorithms. An Evolutionary Computation Approach.
Natural Computing Series. Springer, 2010.
68

[239] G. Paris, D. Robilliard, and C. Fonlupt.
Applying boosting techniques to genetic programming.
In *Artificial Evolution 2001*, volume 2310 of *LNCS*, pages 267–278. Springer, 2001.
66

[240] F.B. Pereira and E. Costa.
Understanding the role of learning in the evolution of busy beaver: A comparison between the Baldwin Effect and Lamarckian strategy.
In *Proc. of the Genetic and Evol. Computation Conf. (GECCO–2001)*, pages 884–891, 2001.
41

[241] Christiaan Perneel and Jean-Marc Themlin.
Optimization of fuzzy expert systems using genetic algorithms and neural networks.
IEEE Trans. on fuzzy systems, 3(3):301–312, 1995.
178

[242] J. Peters and S. Schaal.
Natural actor-critic.
Neurocomputing, 71(7–9):1180–1190, 2008.
157

[243] D.T. Pham and D. Karaboga.
Optimum design of fuzzy logic controllers using genetic algorithms.
J. Systems Eng, 1:114–118, 1991.
181

[244] R. Poli, W.B. Langdon, and N.F. McPhee.
A field guide to genetic programming, freely available at http://www.gp-field-guide.org.uk.
lulu.com, 2008.
58, 71

[245] W.F. Punch, E.D. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody.
Further research on feature selection and classification using genetic algorithms.
In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA93)*, pages 557–564. Morgan Kaufmann, 1993.
54

[246] Amr Radi and Riccardo Poli.
Discovering efficient learning rules for feedforward neural networks using genetic programming.
In Ajith Abraham, Lakhmi Jain, and Janusz Kacprzyk, editors, *Recent Advances in Intelligent Paradigms and Applications*, pages 133–159. Springer Verlag, 2003.
98

[247] M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn, and A.K. Jain.
Dimensionality reduction using genetic algorithms.
IEEE Transactions on Evolutionary Computation, 4(2):164–171, 2000.
53, 54

[248] Rick L. Riolo.
Lookahead planning and latent learning in a classifier system.
In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior (SAB-90)*, pages 316–326. The MIT Press, 1991.
129

[249] R.L. Rivest.
Learning decision lists.
Machine Learning, 2(3):229–246, 1987.
121

[250] S. Romaniuk.
Towards minimal network architectures with evolutionary growth networks.
In *Proc. IEEE Int. Conf. on NNs, IEEE World Congress on Computational Intelligence*, volume 3, pages 1710–1713. IEEE, 1994.
55

[251] S. E. Rouwhorst and A. P. Engelbrecht.
Searching the forest: Using decision trees as building blocks for evolutionary search in classification databases.
In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC00)* [144], pages 633–638.
64

[252] Grzegorz Rozenberg, Thomas Bäck, and Joost Kok, editors.
Handbook of Natural Computing: Theory, Experiments, and Applications.
Springer Verlag, 2010.
40

[253] D. Ruta and B. Gabrys.
Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting.
In J. Kittler and F. Roli, editors, *Proc. 2nd International Workshop on Multiple Classifier Systems*, volume 2096 of *LNCS*, pages 399–408. Springer–Verlag, 2001.
See Kuncheva2004a p.321.
81

[254] L. Sánchez and I. Couso.
Advocating the use of imprecisely observed data in genetic fuzzy systems.
IEEE Transactions on Fuzzy Systems, 15(4):551–562, 2007.
177

[255] R. Santos, J.C. Nievola, and A.A. Freitas.
Extracting comprehensible rules from neural networks via genetic algorithms.
In *Proc. 2000 IEEE Symp. on Combinations of Evolutionary Computation and Neural Networks (ECNN-2000)*, pages 130–139. IEEE, 2000.
55

[256] T. Sasaki and M. Tokoro.
Adaptation toward changing environments: Why darwinian in nature?
In P. Husbands and I. Harvey, editors, *Proceedings of the 4th European conference on artificial life*, pages 145–153. MIT Pess, 1997.
41

[257] Shaun Saxon and Alwyn Barry.
XCS and the Monk's Problems.
In Lanzi et al. [190], pages 223–242.
166

[258] J. David Schaffer, editor.
Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA-89), George Mason University, June 1989. Morgan Kaufmann.
194, 231, 241

[259] Jürgen Schmidhuber.
Evolutionary principles in self-referential learning. (On learning how to learn: The meta-meta-... hook.).
PhD thesis, Institut f. Informatik, Tech. Univ. Munich, 1987.
68

[260] Dale Schuurmans and Jonathan Schaeffer.
Representational Difficulties with Classifier Systems.
In Schaffer [258], pages 328–333.
116

[261] A.J.C. Sharkey.
On combining artificial neural nets.
Connection Science, 8(3–4):299–313, 1996.
74

[262] P.K. Sharpe and R.P. Glover.
Efficient ga based techniques for classification.
Applied Intelligence, 11:277–284, 1999.
53

[263] K. Sirlantzis, M.C. Fairhurst, and M.S. Hoque.
Genetic algorithms for multi-classifier system configuration: a case study in character recognition.
In J. Kittler and F. Roli, editors, *Proc. 2nd International Workshop on Multiple Classifier Systems*, volume 2096 of *LNCS*, pages 99–108. Springer–Verlag, 2001.
See Kuncheva2004a p.321.
81

[264] J.E. Smith.
Coevolving memetic algorithms: A review and progress report.
IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 37(1):6–17, 2007.
40

[265] M.G. Smith and L. Bull.
Genetic programming with a genetic algorithm for feature construction and selection.
Genetic Programming and Evolvable Machines, 6(3):265–281, 2005.
54

[266] Robert E. Smith.
A Report on The First International Workshop on Learning Classifier Systems (IWLCS-92).
NASA Johnson Space Center, Houston, Texas, Oct. 6-9.
ftp://lumpi.informatik.uni-dortmund.de/pub/LCS/papers/lcs92.ps.gz or from ENCORE, The Electronic Appendix to the Hitch-Hiker's Guide to Evolutionary Computation (ftp://ftp.krl.caltech.edu/pub/EC/Welcome.html) in the section on Classifier Systems, 1992.
110

[267] Robert E. Smith.
Memory Exploitation in Learning Classifier Systems.
Evolutionary Computation, 2(3):199–220, 1994.
132, 133

[268] Robert E. Smith and H. Brown Cribbs.
Is a Learning Classifier System a Type of Neural Network?
Evolutionary Computation, 2(1):19–36, 1994.
88

[269] Robert E. Smith and H. Brown Cribbs.
Is a Learning Classifier System a Type of Neural Network?
Evolutionary Computation, 2(1):19–36, 1994.
122

[270] Robert E. Smith and David E. Goldberg.
Variable default hierarchy separation in a classifier system.
In Gregory J. E. Rawlins, editor, *Proceedings of the First Workshop on Foundations of Genetic Algorithms*, pages 148–170, San Mateo, July 15–18 1991. Morgan Kaufmann.
120

[271] Robert E. Smith and H. B. Cribbs III.
Combined biological paradigms.
Robotics and Autonomous Systems, 22(1):65–74, 1997.
88, 122

[272] D. Song, M.I. Heywood, and A.N. Zincir-Heywood.
Training genetic programming on half a million patterns: an example from anomaly detection.
IEEE Transactions on Evolutionary Computation, 9(3):225–239, 2005.
66

[273] N. Srinivas and K. Deb.
Multi-objective function optimization using non-dominated sorting genetic algorithm.
Evolutionary Computation, 2(3):221–248, 1994.
102

[274] Peter Stagge.
Averaging efficiently in the presence of noise.
In *Parallel problem solving from nature*, volume 5, pages 188–197, 1998.
47

[275] Wolfgang Stolzmann.
Learning classifier systems using the cognitive mechanism of anticipatory behavioral control, detailed version.
In *Proceedings of the First European Workshop on Cognitive Modelling*, pages 82–89. Berlin: TU, 1996.
129, 152

[276] Wolfgang Stolzmann.
An Introduction to Anticipatory Classifier Systems.
In Lanzi et al. [190], pages 175–194.
129

[277] Chris Stone and Larry Bull.
For real! XCS with continuous-valued inputs.
Evolutionary Computation, 11(3):298–336, 2003.
117

[278] R. Storn and K. Price.
Minimizing the real functions of the icec'96 contest by differential evolution.
In *Proc. of the IEEE Int. Conf. on Evolutionary Computation*, pages 842–844. IEEE, 1996.
102

[279] M. Stout, J. Bacardit, J.D. Hirst, and N. Krasnogor.
Prediction of recursive convex hull class assignment for protein residues.
Bioinformatics, 24(7):916–923, 2008.
53

[280] R.S. Sutton.
Two problems with backpropagation and other steepest-descent learning procedures for networks.
In *Proc. 8th Annual Conf. Cognitive Science Society*, pages 823–831. Erlbaum, 1986.
89

[281] T. Sziranyi.
Robustness of cellular neural networks in image deblurring and texture segmentation.
Int. J. Circuit Theory App., 24(3):381–396, 1996.
87

[282] A. Tamaddoni-Nezhad and S.H. Muggleton.
Searching the subsumption lattice by a genetic algorithm.
In J. Cussens and A. Frisch, editors, *Proceedings of the 10th International Conference on Inductive Logic Programming*, pages 243–252. Springer-Verlag, 2000.
55

[283] Alireza Tamaddoni-Nezhad and Stephen Muggleton.
A Genetic Algorithms Approach to ILP.
In *Inductive Logic Programming*, volume 2583/2003 of *LNCS*, pages 285–300. Springer, 2003.
55

[284] K. Tharakannel and D. Goldberg.
XCS with average reward criterion in multi-step environment.
Technical report, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2002.
158

[285] S. Thompson.
Pruning boosted classifiers with a real valued genetic algorithm.
In *Research and Development in Expert Systems XV – Proceedings of ES'98*, pages 133–146. Springer, 1998.
55, 78

[286] S. Thompson.
Genetic algorithms as postprocessors for data mining.
In *Data Mining with Evolutionary Algorithms: Research Directions – Papers from the AAAI Workshop. Tech report WS–99–06*, pages 18–22. AAAI Press, 1999.
55, 78

[287] P. Thrift.
Fuzzy logic synthesis with genetic algorithms.
In Lashon B. Booker and Richard K. Belew, editors, *Proceedings of 4th international conference on genetic algorithms (ICGA'91)*, pages 509–513. Morgan Kaufmann, 1991.
181

[288] Andy Tomlinson.
Corporate Classifier Systems.
PhD thesis, University of the West of England, 1999.
133

[289] Andy Tomlinson and Larry Bull.
A Corporate Classifier System.
In A. E. Eiben, T. Bäck, M. Shoenauer, and H.-P. Schwefel, editors, *Proceedings of the Fifth International Conference on Parallel Problem Solving From Nature – PPSN V*, number 1498 in LNCS, pages 550–559. Springer Verlag, 1998.
133

[290] Andy Tomlinson and Larry Bull.
An accuracy-based corporate classifier system.
Journal of Soft Computing, 6(3–4):200–215, 2002.
133

[291] T.H. Tran, C. Sanza, Y. Duthen, and T.D. Nguyen.
XCSF with computed continuous action.
In *Genetic and evolutionary computation conference (GECCO 2007)*, pages 1861–1869. ACM, 2007.
122

[292] K. Tumer and J. Ghosh.
Analysis of decision boundaries in linearly combined neural classifiers.
Pattern Recognition, 29(2):341–348, 1996.
73

[293] Peter Turney.
How to shift bias: Lessons from the baldwin effect.
Evolutionary Computation, 4(3):271–295, 1996.
45

[294] Giorgio Valentini and Francesco Masulli.
Ensembles of learning machines.
In *WIRN VIETRI 2002: Proceedings of the 13th Italian Workshop on Neural Nets-Revised Papers*, pages 3–22. Springer-Verlag, 2002.
76

[295] Manuel Valenzuela-Rendón.
The Fuzzy Classifier System: a Classifier System for Continuously Varying Variables.
In Booker and Belew [30], pages 346–353.
168, 181

[296] Manuel Valenzuela-Rendón.
Reinforcement learning in the fuzzy classifier system.
Expert Systems Applications, 14:237–247, 1998.
168

[297] R. Vallim, D. Goldberg, X. Llorà, T. Duque, and A. Carvalho.
A new approach for multi-label classification based on default hierarchies and organizational learning.
In *Proceedings of the Genetic and Evolutionary Computation Conference, Worrkshop Sessions: Learning Classifier Systems*, pages 2017–2022, 2003.
120

[298] Leonardo Vanneschi and Riccardo Poli.
Genetic programming: Introduction, applications, theory and open issues.
In Grzegorz Rozenberg, Thomas Bäck, and Joost Kok, editors, *Handbook of Natural Computing: Theory, Experiments, and Applications*. Springer Verlag, 2010.
57, 59, 69, 71

[299] G. Venturini.
SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts.
In P.B. Brazdil, editor, *ECML-93 - Proc. of the European Conference on Machine Learning*, pages 280–296. Springer-Verlag, 1993.
37, 151

[300] R. Vilalta and Y. Drissi.
A perspective view and survey of meta-learning.
Artificial Intelligence Review, 18(2):77–95, 2002.
29

[301] A. Wada, K. Takadama, K. Shimohara, and O. Katai.
Learning classifier systems with convergence and generalization.
In L. Bull and T. Kovacs, editors, *Foundations of learning classifier systems*, pages 285–304. Springer, 2005.
161

[302] Atsushi Wada, Keiki Takadama, and Katsunori Shimohara.
Counter example for Q-bucket-brigade under prediction problem.
In *GECCO Workshops 2005*, pages 94–99. ACM Press, 2005.
161

[303] Atsushi Wada, Keiki Takadama, and Katsunori Shimohara.
Learning classifier system equivalent with reinforcement learning with function approximation.
In *GECCO Workshops 2005*, pages 92–93. ACM Press, 2005.
161

[304] Atsushi Wada, Keiki Takadama, and Katsunori Shimohara.
Counter Example for Q-Bucket-Brigade Under Prediction Problem.
In Tim Kovacs, Xavier LLòra, Keiki Takadama, Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Learning Classifier Systems. International Workshops, IWLCS 2003-2005, Revised Selected Papers*, volume 4399 of *LNCS*, pages 128–143. Springer, 2007.
161

[305] Shimon Whiteson and Peter Stone.
Evolutionary function approximation for reinforcement learning.
J. Mach. Learn. Res., 7:877–917, 2006.
41, 43, 47

[306] D. Whitley, T. Starkweather, and C. Bogart.
Genetic algorithms and neural networks: Optimizing connections and connectivity.
Parallel Comput., 14(3):347–361, 1990.
89

[307] Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors.
Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000). Morgan Kaufmann, 2000.
198, 199

[308] Darrell Whitley, V. Scott Gordon, and Keith Mathias.
Lamarckian evolution, the Baldwin effect and function optimization.
In *Parallel Problem Solving from Nature (PPSN-III)*, pages 6–15. Springer-Verlag, 1994.
41

[309] Jason R. Wilcox.
Organizational Learning within a Learning Classifier System.
Master's thesis, University of Illinois, 1995.
Also Technical Report No. 95003 IlliGAL.
31, 34

[310] S. W. Wilson.
Mining oblique data with XCS.
In P.L. Lanzi, W. Stolzmann, and S.W. Wilson, editors, *Advances in learning classifier systems, third international workshop, IWLCS 2000*, volume 1996 of *LNCS*, pages 158–176. Springer, 2001.
117

[311] Stewart W. Wilson.
Bid competition and specificity reconsidered.
Complex Systems, 2:705–723, 1989.
120

[312] Stewart W. Wilson.
ZCS: A Zeroth Level Classifier System.
Evolutionary Computation, 2(1):1–18, 1994.
132

[313] Stewart W. Wilson.
Classifier Fitness Based on Accuracy.
Evolutionary Computation, 3(2):149–175, 1995.
116, 135, 141, 145, 148, 151, 156, 158

[314] Stewart W. Wilson.
Generalization in the XCS classifier system.
In Koza et al. [166], pages 665–674.
141

[315] Stewart W. Wilson.
Get real! XCS with continuous-valued inputs.
In L. Booker, Stephanie Forrest, M. Mitchell, and Rick L. Riolo, editors, *Festschrift in Honor of John H. Holland*, pages 111–121. Center for the Study of Complex Systems, 1999.
117

[316] Stewart W. Wilson.
Mining Oblique Data with XCS.
In *Proceedings of the International Workshop on Learning Classifier Systems (IWLCS-2000), in the Joint Workshops of SAB 2000 and PPSN 2000*, 2000.
Extended abstract.
166

[317] Stewart W. Wilson.
Function approximation with a classifier system.
In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 974–981, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.
159

[318] Stewart W. Wilson.
Classifiers that approximate functions.
Natural Computing, 1(2–3):211–234, 2002.
159

[319] Stewart W. Wilson.
Compact Rulesets from XCSI.
In Lanzi et al. [192], pages 196–208.
148

[320] Stewart W. Wilson.
Three architectures for continuous action.
In Tim Kovacs, Xavier LLòra, Keiki Takadama, Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Learning Classifier Systems. International Workshops, IWLCS 2003-2005, Revised Selected Papers*, volume 4399 of *LNCS*, pages 239–257. Springer, 2007.
122

[321] Stewart W. Wilson.
Classifier conditions using gene expression programming.
In Jaume Bacardit, Ester Bernadó-Mansilla, Martin Butz, Tim Kovacs, Xavier Llorà, and Keiki Takadama, editors, *Learning Classifier Systems. 10th and 11th International Workshops (2006-2007)*, volume 4998/2008 of *Lecture Notes in Computer Science*, pages 206–217. Springer, 2008.
122

[322] Stewart W. Wilson and David E. Goldberg.
A Critical Review of Classifier Systems.
In Schaffer [258], pages 244–255.
133, 166

[323] M.L. Wong and K.S. Leung.
Data mining using grammar based genetic programming and applications.
Kluwer, 2000.
38, 71

[324] K. Woods, W. Kegelmeyer, and K. Bowyer.
Combination of multiple classifiers using local accuracy estimates.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 19:405–410, 1997.
73

[325] John R. Woodward.
GA or GP? That is not the question.
In *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 1056–1063. IEEE, 2003.
57

[326] K. Yamasaki and M. Sekiguchi.
Clear explanation of different adaptive behaviors between Darwinian population and Larmarckian population in changing environment.
In *Proc. Fifth Int. Symp. on Artificial Life and Robotics*, pages 120–123, 2000.
41

[327] X. Yao.
Evolving artificial neural networks.
Proceedings of the IEEE, 87(9):1423–1447, 1999.
40, 86, 87, 91, 92, 96, 97, 98, 104, 107

[328] X. Yao and M.M. Islam.
Evolving artificial neural network ensembles.
IEEE Computational Intelligence Magazine, 3(1):31–42, 2008.
83, 99, 107

[329] X. Yao and Y. Liu.
A new evolutionary system for evolving artificial neural networks.
IEEE Trans. Neural Networks, 8:694–713, 1997.
95, 99

[330] X. Yao and Y. Liu.
Making use of population information in evolutionary artificial neural networks.
IEEE Transactions on Systems, Man and Cybernetics B, 28(3):417–425, 1998.
100

[331] Zhanna V. Zatuchna.
AgentP: a learning classifier system with associative perception in maze environments.
PhD thesis, University of East Anglia, 2005.
129, 152

[332] Z.V. Zatuchna.
AgentP model: Learning Classifer System with Associative Perception.
In *8th Parallel Problem Solving from Nature International Conference (PPSN VIII)*, pages 1172–1182, 2004.
152

[333] B.-T. Zhang and G. Veenker.
Neural networks that teach themselves through genetic discovery of novel examples.
In *Proc. 1991 IEEE Int. Joint Conf. on Neural Networks (IJCNN'91)*, volume 1, pages 690–695. IEEE, 1991.
55