Learning Classifier Systems

From Principles to Modern Systems

Anthony Stein[1] **Masaya Nakata**[2]

[1] **University of Hohenheim, Germany**
anthony.stein@uni-hohenheim.de

[2] **Yokohama National University, Japan**
nakata-masaya-tb@ynu.ac.jp

http://gecco-2020.sigevo.org/

GECCO '20 Companion, July 8–12, 2020

ACM ISBN 978-1-4503-7127-8/20/07.
https://doi.org/10.1145/3377929.3389860

Instructors

Anthony Stein is a tenure track professor at the University of Hohenheim, where he heads the Artificial Intelligence in Agricultural Engineering lab. He received his bachelor's degree (B.Sc.) in Business Information Systems from the University of Applied Sciences Augsburg in 2012. He then moved on to the University of Augsburg for his master's degree (M.Sc.) in computer science with a minor in information economics which he received in 2014. Since November 2019, he also holds a doctorate (Dr. rer. nat.) in computer science from the University of Augsburg. His research is concerned with the application of AI methodology and evolutionary machine learning algorithms to complex self-adaptive and self-organizing (SASO) systems. Dr. Stein is involved in the organization of workshops on intelligent systems and evolutionary machine learning. He serves as reviewer for international conferences and journals, including ACM GECCO or IEEE T-EVC.

Masaya Nakata is an associate professor at Faculty of Engineering, Yokohama National University, Japan. He received his Ph.D. degree in informatics from the University of Electro-Communications, Japan, in 2016. He has been working on Evolutionary Rule-based Machine Learning, Reinforcement Learning, Data mining, more specifically, Learning Classifier System. His contributions have been published as more than 10 journal papers and more than 20 conference papers, e.g., CEC, GECCO, PPSN. He was an organizing committee member of International Workshop on Learning Classifier Systems 2015-2016, 2018-2020 in GECCO conference.

What this tutorial is NOT!

- A comprehensive introduction to the huge field of LCS
- A review of all existent applications of LCS
- A in-depth comparison of Michigan vs. Pittsburgh LCS
- A complete introduction to the theory behind LCS
  → But, we indeed will have a first look ☺

What this tutorial actually is

- An attempt to get the audience in touch with LCS
- An illustrative introduction to make the LCS concept graspable
- A `simplification' to gain an intuition about the overarching learning framework which LCS provide
- A starting point to further dive into the broad field around LCS
- Therefore it is explicitly noted that…
  - we restrict ourselves to Michigan-style LCS
  - we see abstracted views of particular technical details
  - at the end corresponding references for a `deeper dive' are given

Course Agenda

- Introduction
  - A Brief Definition
  - Why LCS?
  - Looking Back: LCS History
- Michigan-style Learning Classifier Systems
  - Building Blocks of LCS
  - Putting it together: A generic LCS
  - Bridging the Gap: Approaching XCS
- XCS Theory in a Nutshell
  - An Overview of Formal Theory Behind LCS
  - Learning Optimality Theory
- Modern Systems
  - XCSF: Piece-wise Online Function Approximation
  - ExSTraCS: Large-scale Supervised Classification
- Summary & Conclusions
  - A Different Perspective
  - Why LCS?
  - Resources & Current Research

Course Agenda

- **Introduction**
  - A Brief Definition
  - Why LCS?
  - Looking Back: LCS History
- Michigan-style Learning Classifier Systems
  - Building Blocks of LCS
  - Putting it together: A generic LCS
  - Bridging the Gap: Approaching XCS
- XCS Theory in a Nutshell
  - An Overview of Formal Theory Behind LCS
  - Learning Optimality Theory
- Modern Systems
  - XCSF: Piece-wise Online Function Approximation
  - ExSTraCS: Large-scale Supervised Classification
- Summary & Conclusions
  - A Different Perspective
  - Why LCS?
  - Resources & Current Research

Introduction

A Brief Definition of Learning Classifier Systems

Learning Classifier Systems (LCS) comprise a family of *flexible*, *evolutionary*, *rule-based machine learning* systems which involve a unique tandem of *local learning* and *global evolutionary optimization* of the collective models' localities.

- **Flexible**
  - Applicability: Have proven successful in a vast variety of domains
  - Extensibility: Define more a framework rather than a specific algorithm
- **Evolutionary**
  - Steady-state Niche Genetic Algorithm (GA) at their heart
  - Neo-Darwinian Survival-of-the-Fittest Principle: Selection, Recombination, Mutation Operators
- **Rule-based**
  - Knowledge is represented via *IF*(condition)-*THEN*(action) rules (aka `classifiers')
  - Divide-and-Conquer: Rules partition the problem space and solve it collectively
- **Machine Learning**
  - Rules/Classifiers, i.e., their internal parameters are learnt via stochastic gradient-based algorithms (Widrow-Hoff delta rule, Recursive Least Squares (RLS), etc.)
  - Capable of Reinforcement Learning (RL), Supervised Learning (SL) and Unsupervised Learning (UL) with only minor and straight-forward changes necessary
  - Thus, applicable to Sequential Problems, Classification, Regression, Clustering

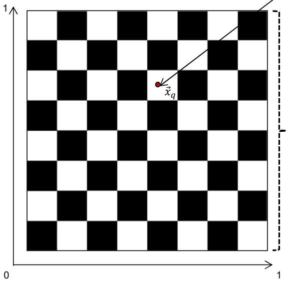
Introduction

Why Learning Classifier Systems? (1/3)

- Interpretability by design
  - Knowledge represented by IF-THEN rules
  - Allows for explicit injection of expert knowledge
- Complexity reduction by design
- Online adaptivity to dynamic learning environments
- Inherent pressures toward generalization
- They are very cool ;-)
- Overarching framework
  - Nearly any kind of ML algorithm can be integrated
- Comparative studies confirm competitive performance

→ Rich body of problem domain and application work in over 40 years of research!

Example Problem
Checkerboard Classification



Task:
Of which shade is the field encompassing the query point $\vec{x}_q$?

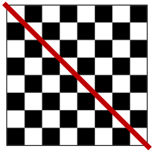
$n_d = 8$ divisions for each dimension with alternating field colors (black/white)

$n = 2$ dimensions each within $[0,1]$

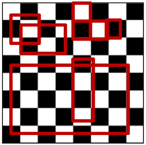
$\vec{x}_q \in [0,1]^2$

Example Problem
Checkerboard Classification

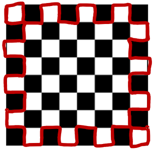
Linearly separable?
→ e.g., Linear Model, Perceptron



Problem Space Partitioning
→ LCS!



Non-linearly separable?
→ e.g., Multi-layer Perceptron



Introduction
Why Learning Classifier Systems? (2/3)

Investigated Problem Domains

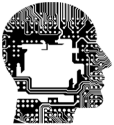
- Adaptive Control (continuous and episodic)
- Uncertain Environments (Noise, Partial Observability)
- Dynamic Environments (Concept Drift/Shift)
- Data Imbalance
  - Class Imbalance
  - Sparsity regarding payoff
- High Dimensionality / Scalability
  - Exploration guidance via expert knowledge
  - Transfer Learning approaches
  - Dimensionality reduction via Autoencoders
- Complexity of underlying problem
  - Heterogeneity, Epistasis
  - Obliqueness, Curvature, Modality, etc.

Introduction

Why Learning Classifier Systems? (3/3)

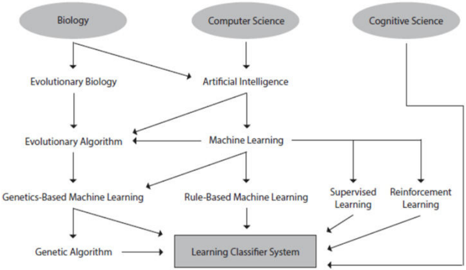
Fields of Real World Application

- Gas-Pipeline Control
- Autonomous Robotics
- Robotic Kinematics
- Motion Control
- Genetics
- Biomedical Knowledge Discovery
- Medical Diagnosis
- Cognitive Modeling
- Traffic Control
- Smart Camera Networks
- Games
- … and many more!



Introduction
Looking Back: History of LCS*

- Learning Classifier System (LCS)
  - In retrospect, an odd name.
  - There are many machine learning systems that learn to classify but are not LCS algorithms.
  - E.g. Decision trees
- Also referred to as…
  - Rule-Based Machine Learning (RBML)
  - Genetics Based Machine Learning (GBML)
  - Adaptive Agents
  - Cognitive Systems
  - Production Systems
  - Classifier System (CS, CFS)



* Image adapted from [49]

* Adapted from Urbanowicz's previous tutorials

Introduction

Looking Back: History of LCS*

1970's ➡

- Genetic algorithms and CS-1 emerge
- Research flourishes, but application success is limited.

1980's

1990's

2000's

2010's

- LCSs are one of the earliest artificial cognitive systems – developed by **John Holland (1978)** [14].
  - His work at the University of Michigan introduced and popularized the genetic algorithm.
- Holland's Vision: **Cognitive System One (CS-1)**
  - Fundamental concept of classifier rules and matching.
  - Combining a credit assignment scheme with rule discovery.
  - Function on environment with infrequent payoff/reward.
- The early work was ambitious and broad. This has led to many paths being taken to develop the concept over the following 40 years.
- CS-1 archetype would later become the basis for **`Michigan-style' LCSs**.

* Adapted from Urbanowicz's previous tutorials

Introduction

Looking Back: History of LCS*

1970's

- Pittsburgh-style algorithms introduced by **Smith** in Learning Systems One (LS-1) [35]

1980's

- LCS subtypes appear: Michigan-style vs. Pittsburgh-style
- Holland adds reinforcement learning to his system.
- Term `Learning Classifier System' adopted.
- Research follows Holland's vision with limited success.
- Interest in LCS begins to fade.

1990's

- **Booker** suggests niche-acting GA (in [M]) [5]
- **Holland** introduces bucket brigade credit assignment [15]
- Interest in LCS begins to fade due to inherent algorithm complexity and failure of systems to behave and perform reliably

2000's

2010's

* Adapted from Urbanowicz's previous tutorials

Introduction

Looking Back: History of LCS*

1970's

1980's

1990's

2000's

2010's

- **Frey & Slate** present an LCS with predictive accuracy fitness rather than payoff-based strength [11]
- **Riolo** introduces CFCS2, setting the scene for Q-learning like methods and anticipatory LCSs [34]
- **Wilson** introduces simplified LCS architecture with his Zeroth-level Classifier System (ZCS), a strength-based system [59]

- REVOLUTION!
- Simplified LCS algorithm architecture with ZCS
- XCS is born: First reliable and more comprehensible LCS
- First classification and robotics applications (real-world)

- **Wilson** revolutionizes LCS algorithms with accuracy-based rule fitness in his XCS Classifier System (XCS) [60]
- **Holmes** applies LCS to problems in epidemiology [16]
- **Stolzmann** introduces Anticipatory Classifier Systems (ACS) [44]

* Adapted from Urbanowicz's previous tutorials

Introduction

Looking Back: History of LCS*

1970's

1980's

1990's

2000's

2010's

- **Wilson** introduces XCS for function approximation (XCSF) [64]
- **Kovacs** explores a number of practical and theoretical LCS questions [21,22]
- **Bernadó-Mansilla** introduce sUpervised Classifier System (UCS) for supervised learning [4]
- **Bull** explores LCS theory in simple systems [6]
- **Bacardit** introduces two Pitt-style LCS systems GAssist and BioHEL with emphasis on data mining and improved scalability to larger datasets [1,2]
- **Holmes** introduces EpiXCS for epidemiological learning. Paired with the first LCS graphical user interface to promote accessibility and ease of use [17]
- **Butz** introduces first online learning visualization for function approximation
- **Lanzi & Loiacono** explore computed actions

- LCS algorithm specializing in supervised learning and data mining start appearing
- LCS scalability becomes a central research theme
- Increasing interest in epidemiological and bioinformatics
- Facet-wise theory and applications

* Adapted from Urbanowicz's previous tutorials

Introduction

Looking Back: History of LCS*

1970's

1980's

1990's

2000's

2010's

- **Franco & Bacardit** explored GPU parallelization of LCS for scalability.
- **Urbanowicz & Moore** introduced statistical and visualization strategies for knowledge discovery in an LCS [53]. Also explored use of `expert knowledge' to efficiently guide GA [55], introduced attribute tracking for explicitly characterizing heterogeneous patterns [54,57].
- **Browne and Iqbal** explore new concepts in reusing building blocks (i.e., code fragments) . Solved the 135-bit multiplexer reusing building blocks from simpler multiplexer problems [19].
- **Bacardit** successfully applied BioHEL to large-scale bioinformatics problems also exploring visualization strategies for knowledge discovery [3].
- **Urbanowicz** introduced ExSTraCS for supervised learning [51,56]. Applied ExSTraCS to solve the 135-bit multiplexer directly.

- Increased interest in supervised learning applications persists.
- Emphasis on solution interpretability and knowledge discovery.
- Scalability improving – 135-bit multiplexer solved!
- GPU interest for computational parallelization.
- Broadening research interest from American & European to include Australasian & Asian.

* Adapted from Urbanowicz's previous tutorials

Introduction

Looking Back: History of LCS*

1970's

1980's

1990's

2000's

2010's

~40 years of LCS research has…

- Clarified understanding.
- Produced algorithmic descriptions.
- Determined 'sweet spots' for run parameters.
- Delivered understandable 'out of the box' code.
- Demonstrated LCS algorithms to be…
  - Flexible
  - Widely applicable
  - Uniquely functional on particularly complex problems.

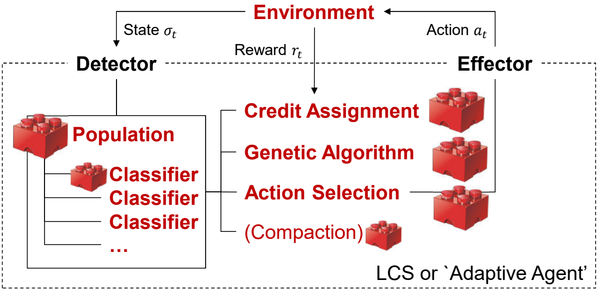
* Adapted from Urbanowicz's previous tutorials

Course Agenda

- Introduction
  - ✓ A Brief Definition
  - ✓ Why LCS?
  - ✓ Looking Back: LCS History
- **Michigan-style Learning Classifier Systems**
  - Building Blocks of LCS
  - Putting it together: A generic LCS
  - Bridging the Gap: Approaching XCS
- XCS Theory in a Nutshell
  - An Overview of Formal Theory Behind LCS
  - Learning Optimality Theory
- Modern Systems
  - XCSF: Piece-wise Online Function Approximation
  - ExSTraCS: Large-scale Supervised Classification
- Summary & Conclusions
  - A Different Perspective
  - Why LCS?
  - Resources & Current Research

Michigan-style LCS

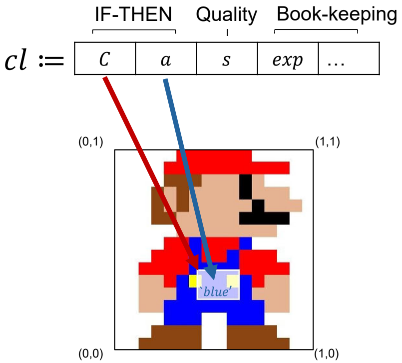
Building Blocks of a Learning Classifier System



Michigan-style LCS
BBs of LCS: Classifier

Classifier cl

- IF-THEN rule
  - Condition $cl.C$
  - Action $cl.a$
- Condition $cl.C$ encodes input subspace $cl.C \subseteq X$
  - Conditions of $cl's$ are not disjoint!
- Rule strength $cl.s$, e.g.,
  - Predicted Payoff
  - Prediction Accuracy
- Book-keeping parameters
  - Experience
  - Niche size
  - Numerosity
  - etc.



`Mario' multi-class problem [41]

* dot-notation denotes reference to parameters of specified classifier cl

Michigan-style LCS

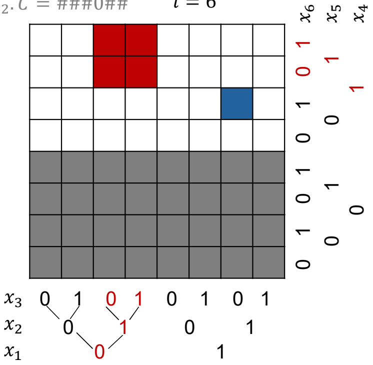
BBs of LCS: Classifier’s Condition

Ternary Encoded Condition

- Encodes schema within problem’s input/state space
- For binary input spaces $\mathbb{B}^l$
- One bit of input instance covered by one symbol in the condition
- Symbol from ternary alphabet
  $$\Sigma = \{0,1,\#\}$$
  - `#’ serves as don’t care / wildcard
- Condition is concatenation of symbols
  - $C \coloneqq (c_1, \ldots, c_l), c_i \in \{0,1,\#\}$
- Condition also encodes chromosome for the GA
- Example Problems:
  - k-Multiplexer, Majority-On, Parity, etc.

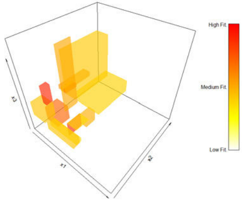
$cl_1.C = 01\#11\#$ $cl_3.C = 110101$

$cl_2.C = \#\#\#0\#\#$ $l = 6$



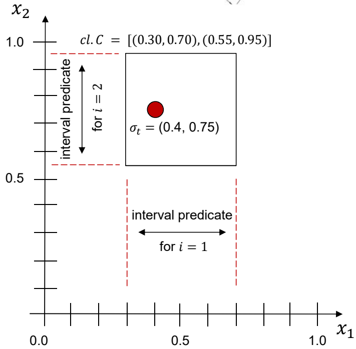
Michigan-style LCS

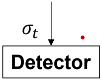
BBs of LCS: Classifier’s Condition



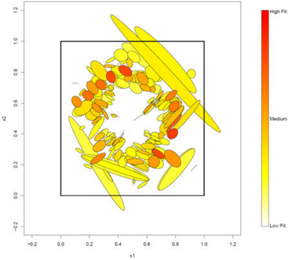
Interval-based Condition

- Encodes subspace within problem’s input/state space
- Real-valued input spaces $\mathbb{R}^d$
- One dimension $i = 1, \dots, d$ of an input instance is covered by one interval predicate in C
  - i-th interval predicate (l_i, u_i)
  - Lower bound l_i, upper bound u_i
  - Ordered vs. unordered Bound
- C is concatenation of intervals
  - $C \coloneqq [(l_1, u_1), \dots, (l_d, u_d)], l_i, u_i \in \mathbb{R}$
- Each bound is one gene in chromosome
- Example inputs:
  - Continuous values e.g., Traffic flows at intersections, Sensory data
  - Nominal (gender, blood group) or ordinal features (age, salary, etc.)



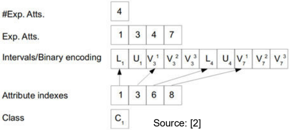


Michigan-style LCS
BBs of LCS: Classifier's Condition

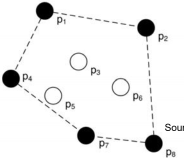


Many more condition alphabets

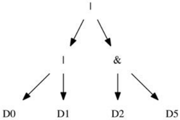
- ❖ Hyperellipsoids (e.g., [9])
  - Covariance Matrix representation
  - Explicit geometric representation
- ❖ S-expressions / Code Fragments [19]
- ❖ Convex Hulls [27]
- ❖ Mixed Discrete-Continuous Attribute List Knowledge Representation (ALKR) [2]
- ❖ Neural Networks [7], etc.



Source: [2]



Source: [27]



Source: [19]

Michigan-style LCS

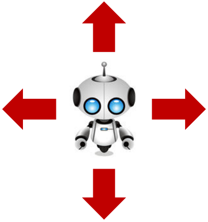
BBs of LCS: Classifier's Action

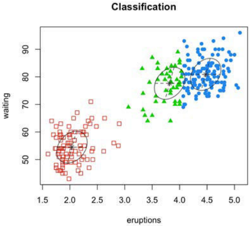
Discrete Actions

- Depends on the learning task
  - Reinforcement Learning: Action
  - Classification: Class/Endpoint
  - Regression: No action needed!
- Examples:
  - Robot navigation: Turn left, right, up, down
  - Medical diagnosis: Tumor is benign or malignant
  - Traffic light control: Signal plan A, B or C
- Large action spaces A
  - Each rule maintains a single action
  - Many rules needed for a complete mapping of the state-action-space
- Continuous Actions
  - Selection turns out difficult
  - But: Approaches do exist

Effector

a_{exec}





Michigan-style LCS

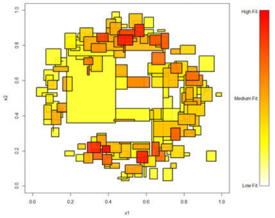
BBs of LCS: Population

Population $[P]$

- ❖ The set of all rules/classifiers
- ❖ Constitutes knowledge base
- ❖ Entirety of $cl \in [P]$ collectively makes up the global model
- ❖ Contains many transient rules
- ❖ Contains $n \leq N$ classifiers
  - N is a critical hyperparameter
  - Single classifier can subsume others → numerosity $cl.num$
  - Size of $[P]$ is limited s.t. $\sum_{cl \in [P]} cl.num \leq N$
- ❖ $[P]$ usually starts `tabula rasa'
- ❖ Can be initialized a priori
  - Randomly
  - Expert Knowledge / Default rules

$[P]$

	IF-THEN		Quality	Book-keeping	
cl_1	C	a	s	exp	…
cl_2	C	a	s	exp	…
cl_3	C	a	s	exp	…
⋮					
cl_n	C	a	s	exp	…



Michigan-style LCS

BBs of LCS: Population

Population $[P]$

- ❖ The set of all rules/classifiers
- ❖ Constitutes knowledge base
- ❖ Entirety of $cl \in [P]$ collectively makes up the global model
- ❖ Contains many transient rules
- ❖ Contains $n \leq N$ classifiers
  - N is a critical hyperparameter
  - Single classifier can subsume others → numerosity $cl.num$
  - Size of $[P]$ is limited s.t. $\sum_{cl \in [P]} cl.num \leq N$
- ❖ $[P]$ usually starts `tabula rasa'
- ❖ Can be initialized a priori
  - Randomly
  - Expert Knowledge / Default rules

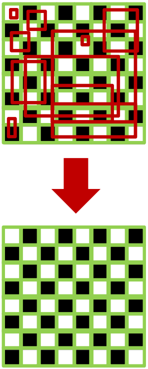
$[P]$

	IF-THEN		Quality	Book-keeping	
cl_1	C	a	s	exp	$\ldots$
cl_2	C	a	s	exp	$\ldots$
cl_3	C	a	s	exp	$\ldots$
			$\vdots$		
cl_n	C	a	s	exp	$\ldots$

Michigan-style LCS
BBs of LCS: Compaction

Distillation of $[P]$

- ❖ Not necessary for learning success!
- ❖ Increases inference speed and comprehensibility of model
- ❖ Removes transient rules from $[P]$
  - →Smaller collection of `predictive‘ rules
- ❖ Different approaches, e.g.,
  - Condensation [60]
  - Greedy compaction [9]
  - Quick Rule Filtering [47]
- ❖ Typically applied at the end of learning or after convergence
- ❖ Up to ~90 % smaller size of $[P]$
- ❖ But only marginal increase in prediction error

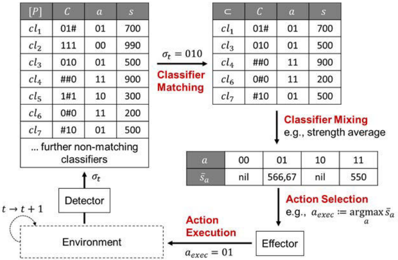


Michigan-style LCS

BBs of LCS: Action Selection

Action Selection

- The actual `inference' step
- Chooses the action/prediction at each time step / for each situation
- Aka Policy $\pi: S \rightarrow A$ (from RL domain)
- More generally referred to as Performance Component
  (1) Classifier Matching → determines niche!
  (2) Classifier Mixing → collective solution!
  (3) Action Selection
  (4) Action Execution
- Handles Exploration vs. Exploitation trade-off, e.g.,
  - Interleaving random/greedy selection
  - ϵ-greedy policy
  - Purely explore and exploit afterwards



* adapted from [39]

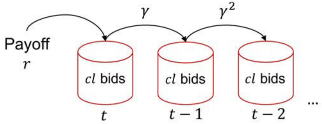
Michigan-style LCS

BBs of LCS: Credit Assignment

Credit Assignment

- Aka **Reinforcement Component**
- Learning comes into play
- Reward signal from environment
  - Immediate reward → may be 0
  - Delayed payoff → goal reached, 1000
- Single-step vs. Multi-step
- Correct / Incorrect Action Selection
- Reward / Punish
- Problem: Long action sequences
- Which classifiers to reinforce / attenuate?
- Early `stage-setting' classifiers
- Adapts selected classifiers' learnable parameters, i.e., strength $cl.s$
- Updates book-keeping parameters

The early algorithm:
(Implicit) Bucket Brigade [15,59]



$$cl.s_t = cl.s_{t-1} - \gamma cl.s_{t-1} + \frac{\gamma}{|[A]_t|} \sum_{cl_j \in [A]_t} cl_j.s_t$$

The modern approach:
Temporal Difference Learning

$$cl.s_t = cl.s_{t-1} + \beta(\underbrace{\underbrace{r_{t-1} + \gamma \max_a \bar{s}_a}_{\text{Immediate reward } r_{t-1} + \text{ current max. strength} \rightarrow \text{ back-up}} - cl.s_{t-1}}_{\text{New estimate – old estimate} \rightarrow \text{ TD}})$$

* Classifiers cl that were in $[A]$ of the previous cycle are updated here!

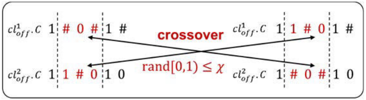
Michigan-style LCS

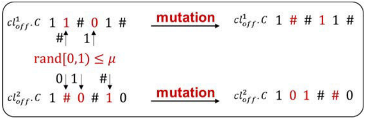
BBs of LCS: Genetic Algorithm

Genetic Algorithm

- Aka **Discovery Component**
- Steady-state Niche GA
- Periodic execution
- Optimizes coverage of the input space
- Usually, only conditions are altered
  - However, action mutation exists
- Fitness measure
  - Strength $cl.s$ in ZCS and older variants
  - Relative accuracy $cl.\kappa'$ in XCS and descendants (XCSF, UCS, ExSTraCS)
- Hyperparameters
  - Mutation rate μ
  - Crossover probability χ
  - Selection mechanism (Roulette-wheel vs. Tournament)
  - GA activation threshold θ_{GA}

Ternary Case





* adapted from [39]

Michigan-style LCS

BBs of LCS: Genetic Algorithm

Genetic Algorithm

- Still, steady-state niche GA
- Still, periodic execution
- Still, optimizes coverage of the input space
- Same fitness measure
- Additional hyperparameter
- Mutation spread m_0

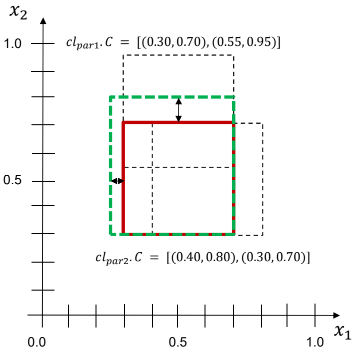
1st offspring after crossover:

$cl_{off1}.C \ = \ [(0.30, 0.70), (0.30, 0.70)]$

1st offspring after mutation:

$cl_{off1}.C \ = \ [(0.25, 0.70), (0.30, 0.80)]$

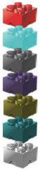
Real-valued case

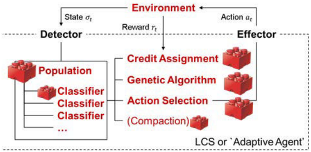


Michigan-style LCS

Putting all together

- Building blocks are the most basic components of LCS
- Each block can have more than one `color‘
- E.g., for credit assignment:
  - Bucket Brigade Algorithm
  - Profit Sharing Plan
  - Implicit Bucket Brigade
  - Q-Learning
  - Widrow-Hoff (single-step)
  - Linear Least Square
  - Recursive Least Square



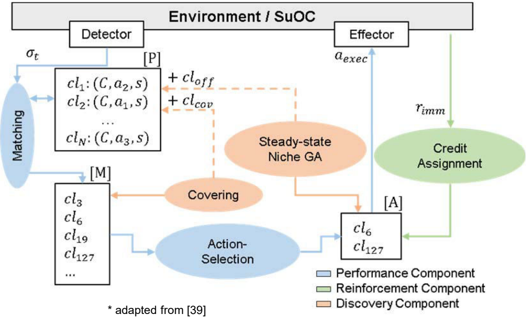


- Select the most promising block for your problem and put it together

- **→ LCS provide a generic framework, not a single algorithm!**

Michigan-style LCS

Putting all together: A Generic LCS



* adapted from [39]

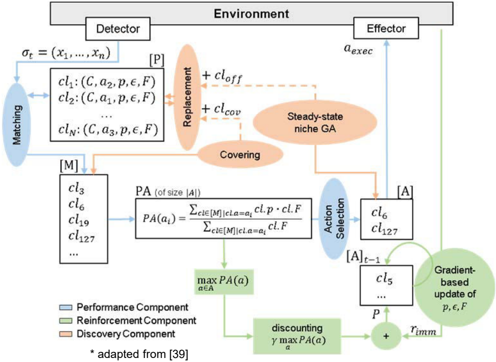
Michigan-style LCS

Bridging the Gap: Approaching XCS

- XCS Classifier System (XCS) [60]
- Due to Stewart W. Wilson
- \`Classifier fitness based on accuracy'
- Replaces strength $cl.s$ with triplet
  - Predicted payoff $cl.p$
  - Prediction error $cl.\epsilon$
  - Fitness $cl.F$
- BBA credit assignment replaced with Q-learning-like update
- Applies niche instead of panmictic GA
  - first on $[M]$ later on $[A]$ instead of $[P]$
- Extension of the Zeroth-level Classifier System (ZCS) [59]

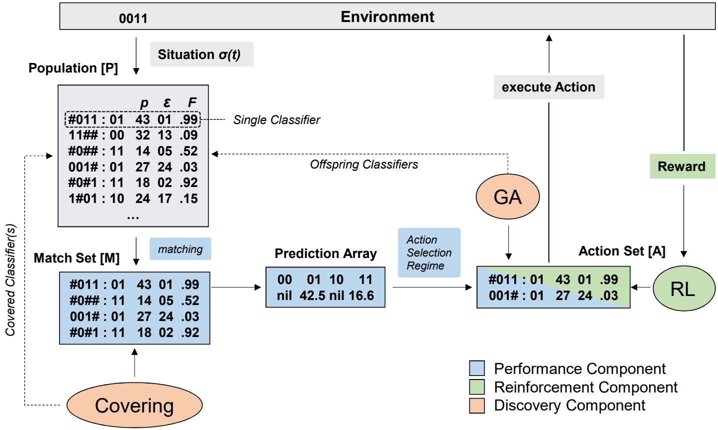
Michigan-style LCS

XCS Classifier System: Overview



Michigan-style LCS

XCS Classifier System: Overview

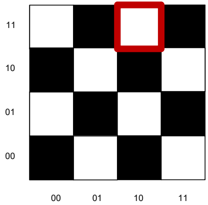


XCS Classifier System

A quick main loop run-through

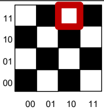
Discrete Checkerboard

- What is the situation $\sigma(t)$?
- The coordinates of the red boxed field (10,11)
- Starting horizontally: $\sigma(t)$=1011
- What are the possible actions $a \in A$?
- `black‘ = 1
- `white‘ = 0
- What payoff can be retrieved?
- 1000 for correct action
- 0 for wrong action



XCS Main Loop
Matching

1011 **Environment**



Situation ***σ(t)***

Population [P]

	p	*ε*	*F*
#011 : 0	43	01	.99
11## : 0	32	13	.09
#0## : 1	14	05	.52
101# : 0	27	24	.03
#0#1 : 1	18	02	.92
1#01 : 1	24	17	.15
…			

Single Classifier

matching

Match Set [M]

#011 : 0	43	01	.99
#0## : 1	14	05	.52
101# : 0	27	24	.03
#0#1 : 1	18	02	.92

XCS Main Loop
Matching

- At each timestep t XCS retrieves a binary string on length $n + m$
- This string is denoted as $\sigma(t) \in \{0,1\}^{n+m}$
- Example for discrete CBP ($n = 2$, $m = 2$ bits per dimension) and $t = 1$: $\sigma(1) = 1011$
- Each classifier maintains a condition C
- The conditions are encoded ternary, i.e. $C \in \{0,1,\#\}^{n+m}$
- The # symbol serves as wildcard or \`don't care' operator
- Examples of conditions: (is matching $\sigma(1)$?)
    - 1#11
    - #011
    - 01#1

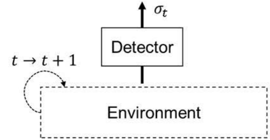
Matching is the process of scanning the entire population $[P]$ for classifiers with a condition that is \`fulfilled' by the situation $\sigma(t)$

XCS Main Loop
Matching: A simple example

$[P]$	C	a	p	ϵ	F
cl_1	01#	01	700	200	0.8
cl_2	111	00	990	110	0.9
cl_3	010	01	500	500	0.5
cl_4	##0	11	900	600	0.1
cl_5	1#1	10	300	500	0.4
cl_6	0#0	11	200	50	0.9
cl_7	#10	01	500	400	0.7
... further non-matching classifiers					

$\xrightarrow{\sigma_t = 010}$ **Classifier Matching**

$[M]$	C	a	p	ϵ	F
cl_1	01#	01	700	200	0.8
cl_3	010	01	500	500	0.5
cl_4	##0	11	900	600	0.1
cl_6	0#0	11	200	50	0.9
cl_7	#10	01	500	400	0.7



* adapted from [39]

XCS Main Loop

System Prediction

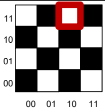
1011 **Environment**

Situation ***σ(t)***

Population [P]

	p	*ε*	*F*
#011 : 0	43	01	.99
11## : 0	32	13	.09
#0## : 1	14	05	.52
101# : 0	27	24	.03
#0#1 : 1	18	02	.92
1#01 : 1	24	17	.15
…			

Single Classifier



Offspring Classifiers

Match Set [M] *matching*

#011 : 0	43	01	.99
#0## : 1	14	05	.52
101# : 0	27	24	.03
#0#1 : 1	18	02	.92

Prediction Array

0	1
42.5	16.6

The calculation of the system prediction is the actual `inference‘ step! Here, the local models are combined (`mixed‘) into a collective target prediction!

XCS Main Loop
System Prediction

- The system prediction $P(a)$ is a fitness-weighted sum of predictions of all classifiers in $[M]$ advocating action a

$$P(a) = \frac{\sum_{cl \in [M] | cl.a = a} cl.F * cl.p}{\sum_{cl \in [M] | cl.a = a} cl.F}$$

- Especially at this place, the separation of strength and accuracy becomes apparent!
- For each possible action $a \in A$ there exists one entry within the PA
- If a is not represented in $[M]$, the PA entry is nil

XCS Main Loop

System Prediction: A simple example

$[P]$	C	a	p	ϵ	F
cl_1	01#	01	700	200	0.8
cl_2	111	00	990	110	0.9
cl_3	010	01	500	500	0.5
cl_4	##0	11	900	600	0.1
cl_5	1#1	10	300	500	0.4
cl_6	0#0	11	200	50	0.9
cl_7	#10	01	500	400	0.7
... further non-matching classifiers					

$\sigma_t = 010$

Classifier Matching

$[M]$	C	a	p	ϵ	F
cl_1	01#	01	700	200	0.8
cl_3	010	01	500	500	0.5
cl_4	##0	11	900	600	0.1
cl_6	0#0	11	200	50	0.9
cl_7	#10	01	500	400	0.7

Classifier Mixing (fitness-based):

$$PA(a_i) = \frac{\sum_{cl \in [M] | cl.a = a_i} cl.p \cdot cl.F}{\sum_{cl \in [M] | cl.a = a_i} cl.F}$$

PA	00	01	10	11
	nil	580	nil	270

Action Selection (greedy):

$$a_{exec} := \underset{a}{\operatorname{argmax}} PA(a)$$

Effector

Action Execution

$a_{exec} = 01$

Environment

$t \rightarrow t+1$

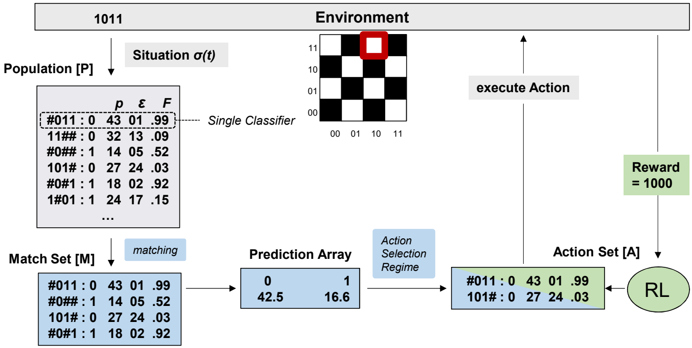
Detector

σ_t

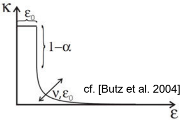
* adapted from [39]

XCS Main Loop

Credit Assignment



XCS Main Loop
Credit Assignment

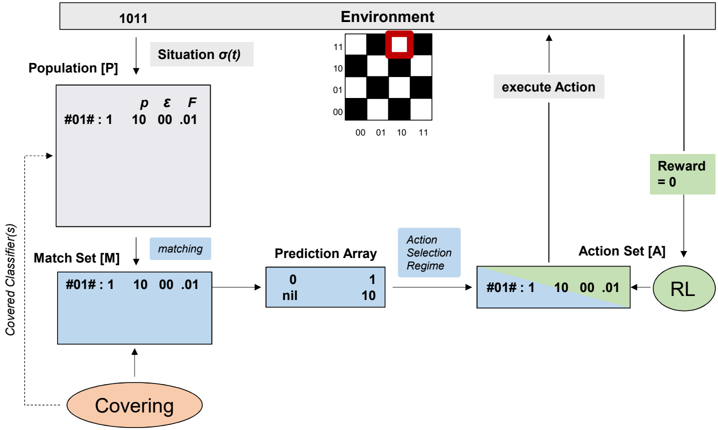


cf. [Butz et al. 2004]

- $\epsilon_j \leftarrow \epsilon_j + \beta\left(\left|P - p_j\right| - \epsilon_j\right)$
- $p_j \leftarrow p_j + \beta(P - p_j)$
- $F_j \leftarrow F_j + \beta\left(\kappa_j' - F_j\right),\ \kappa_j' = \frac{cl_j.\kappa \cdot cl_j.num}{\sum_{cl_i \in [A]} cl_i.\kappa \cdot cl_i.num},\ \kappa_j = \alpha\left(\frac{\epsilon_j}{\epsilon_0}\right)^{-\nu}$

- β is the learning rate (typically set to 0.2)
- α (often set to 0.1) and ν (usually set to 5) control how strong accuracy decreases when error is higher than ϵ_0
- ϵ_0 defines the targeted error level of the system
- In single-step problems, P is set to the immediate reward r_{imm}
- Classifier parameters are updated by means of the Widrow-Hoff (or delta) rule in combination with the moyenne adaptiv modifiée (MAM) technique

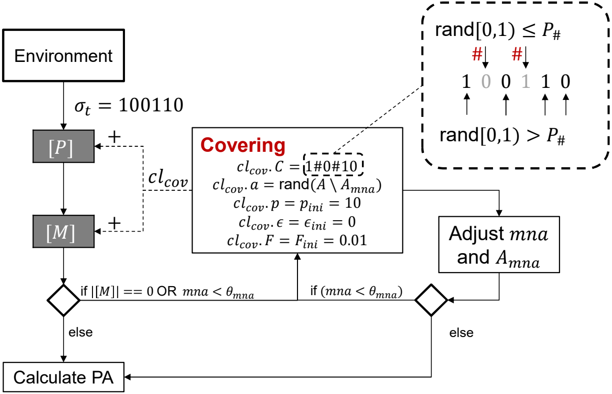
XCS Main Loop
Covering



XCS Main Loop
Covering

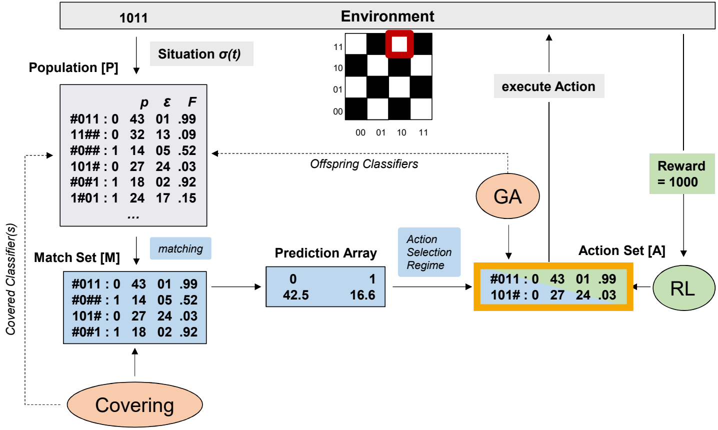
- Covering is the process of generating at least one novel classifier that matches the current input $\sigma(t)$ whenever:
  - Match set $[M]$ is empty (i.e. no matching cl in [P])
  - $[M]$ is poor, i.e. average fitness below a certain threshold
  - $[M]$ contains less then θ_{mna} distinct actions
- The condition of the covered classifier cl_{cov} is initially set to the current input
- Additionally, each bit is replaced by a # (for generalization purposes) with probability $P_{\#}$
- The action is selected equiprobably between actions not present in $[M]$
- Values for p, ϵ and F are set to predefined initial values (typically 10.0, 0.0 and 0.01, respectively)

XCS Main Loop
Covering



* adapted from [39]

XCS Main Loop
Genetic Algorithm



XCS Main Loop

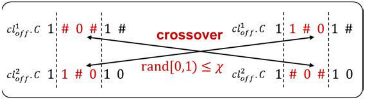
Genetic Algorithm: Invocation and Selection

- One of the most essential parts of XCS is the incorporated steady-state niche GA (steady-state: only a small fraction of the population is replaced)
- It is triggered when the average time over all classifiers in $[A]$ since the last GA invocation is greater than θ_{GA} (often set to 12)
  - $t - \overline{ts} > \theta_{GA}$, where $\overline{ts} = \frac{\sum_{cl \in [A]} cl.ts}{|[A]|}$
- The GA selects two parents from $[A]$ with a probability proportional to their fitness values (roulette-wheel selection)
  - The higher a classifier‘s fitness, the higher the selection chance
- The selected parents are copied to generate two offspring classifiers $cl_{off}^{1}, cl_{off}^{2}$

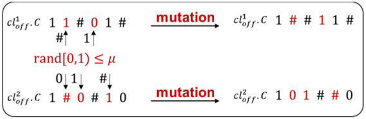
XCS Main Loop

Genetic Algorithm: Crossover and Mutation

- The conditions of both cl_{off} are crossed with probability $\chi = 0.8$ (crossover operator)
  - One-point crossover: Each offspring classifier's condition is split at a certain point and switched with the other offspring classifier
  - n-point crossover: more than one point is determined for switching
  - Uniform crossover: Each value is switched with a certain probability (often 0.5)

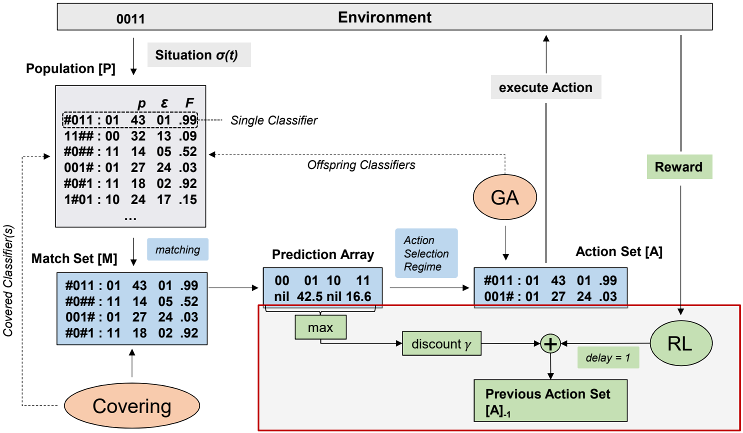


- Afterward, each bit is flipped with probability $\mu = 0.04$ to one of the other allowed alleles (mutation operator)
  - E.g. 2nd bit is set to `1', mutation can flip this bit to `0' or `#'



XCS Main Loop

Sequential Problem Solving (Multi-step)



XCS Main Loop
Sequential Problem Solving (Multi-step)

- r may or may not be retrieved in each step
- One has to distinguish immediate reward (r^{imm}) and total reward or payoff r at the end of a task (e.g. finally food was found)
- Update of classifier attributes is performed on the action set of the previous timestep $t-1$ ($[A]_{-1}$)
- The maximum system prediction $P(a)$ from the current PA is discounted by a factor γ (usually $\gamma = 0.95$)
- Additionally, the immediate reward gained for performing the action in the previous state (of time step $t-1$) r_{t-1}^{imm} is added (may be 0)
- This delay allows to retrieve „information from the future“
- In single-step environments $P = r^{imm}$
- In multi-step problems $P = r_{t-1}^{imm} + \gamma * \max_{a} PA(a)$

XCS Main Loop

Sequential Problem Solving (Multi-step)

- Single-step update of p:

$$p_j \leftarrow p_j + \beta\left(P - p_j\right)$$

- Substituting P yields us the multi-step update formula
- Multi-step update of p:

$$p_j \leftarrow p_j + \beta(r_{t-1}^{imm} + \gamma \max_a PA(a) - p_j)$$

- Do you know this update procedure from anywhere else?

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_a Q(s',a) - Q(s,a)]$$

XCS Main Loop

Multi-step Credit Assignment: A sample calculation

PA	00	01	10	11
	720	nil	360	nil

$\max_{a \in A} PA(a) = 720$

$0.9 \cdot 720 = 648$

discount by γ

Environment

$r_{imm} = 1000$

immediate reward

$P = 1648$

refine attributes using Widrow-Hoff delta rule

$[A]_{-1}$	C	a	p	ϵ	F
cl_1	01#	01	700	200	0.8
cl_3	010	01	500	500	0.5

e.g. p of cl_1:

$$cl_1.p \leftarrow 700 + 0.2\left(1000 + 0.9\max[720{,}360] - 700\right)$$

$$cl_1.p = 889.6$$

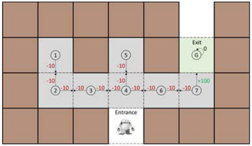
* adapted from [39]

XCS Main Loop

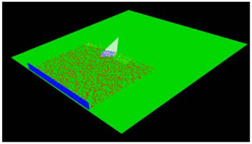
Sequential Problem Solving (Multi-step)

Examples for multi-step environments:

- Animat scenarios:
  - Agent is seeking food / gold / exit / etc.
  - E.g., Woods or Maze scenarios



- Step-wise adjustment of a control variable:
  - Pan, Tilt, Zoom in Smart Camera Networks
  - Mountain Car
  - Inverse Pendulum



- Movement decisions:
  - `Move to beacon‘ minigame in StarCraft II LE

Course Agenda

- Introduction
  - ✓ A Brief Definition
  - ✓ Why LCS?
  - ✓ Looking Back: LCS History
- Michigan-style Learning Classifier Systems
  - ✓ Building Blocks of LCS
  - ✓ Putting it together: A generic LCS
  - ✓ Bridging the Gap: Approaching XCS
- **XCS Theory in a Nutshell** *(presented by Dr. Nakata)*
  - An Overview of Formal Theory Behind LCS
  - Learning Optimality Theory
- Modern Systems
  - XCSF: Piece-wise Online Function Approximation
  - ExSTraCS: Large-scale Supervised Classification
- Summary & Conclusions
  - A Different Perspective
  - Why LCS?
  - Resources & Current Research

XCS Theory in a Nutshell

Much formal work already done!

- One disadvantage of LCS often mentioned is…

“[…] less formal understanding and a
relatively small body of theoretical work […]”

- We should put emphasis on “relatively”
- Sometimes experienced misconception that…

no theory could exist for LCS!

- **This is not true!**

XCS Theory in a Nutshell

An Overview of Formal Theory Behind LCS

Facetwise Theory
- Evolutionary Pressures (Butz et al.)
- Solution growth, sustenance
- Effective search

Hypotheses
- Generalization Hypothesis (Wilson)
- Optimality Hypothesis (Kovacs)
- Theory on Strong Overgenerals (Kovacs)

Hyperparameter derivation
- Parameter bounds (Butz, Stalph)
- Optimal parameters (Nakata et al.)

XCS Theory

GA analysis in XCS
- Markov-chain analysis (Bull, Butz)
- Selection pressure (Butz, Orriols-Puig)
- Selection probability (Kovacs et al.)

Convergence and time bounds
- PAC learnability of k-DNF (Butz)
- Domino convergence model (Butz et al.)
- Convergence proofs under simplifications (Drugowitsch, Wada)

Learning problem analysis
- Imbalanced domains (Orriols-Puig et al.)
- Complexity of classification problems (Bernadó-Mansilla et al.)

Formalizations (models)
- Simple LCSs (Goldberg, Bull)
- Probabilistic model (Drugowitsch & Barry)

*see [33] for a brief survey

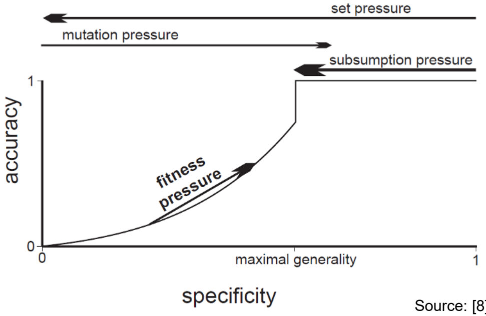
XCS Theory in a Nutshell
Facetwise Approach

- **Facetwise Theory Approach** (due to Goldberg [13])
  - Proposed to analyze and understand GAs
  - Partitioning of a system into its most relevant components
  - Analysis in separation
  - Afterward, combine and investigate interactions
  - Answer questions: ***What***?, ***How***? and ***When***?

- **Facetwise LCS Theory** (due to Butz et al. [8,10])
  - I. Design evolutionary pressures most effectively
    - – Fitness guidance, parameter estimation, generalization
  - II. Ensure solution growth and sustenance
    - – Population initialization, schema supply, growth and sustenance
  - III. Enable effective solution search
    - – Mutation, recombination, local vs. global structure
  - IV. Consider additional challenges in multi-step problems
    - – Effective policy, problem sampling, reward propagation

XCS Theory in a Nutshell

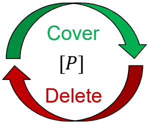
Evolutionary Pressures (or `*How it learns*?')



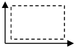
Source: [8]

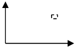
XCS Theory in a Nutshell

Learning Bounds (or `*When it learns*?')



- Main challenges (schema and covering)
  - Covering Challenge
    - Ensure coverage and GA application
    - Prevent being trapped in a covering-deletion-cycle
  - Schema Challenge
    - Ensure that fitness pressure applies
    - From both directions: over-general and over-specific classifiers
    -





- Derived bounds:
  - *Covering* bound
  - *Schema* bound
  - *Reproductive opportunity* bound
  - *Niche support* bound
  - *Learning time* bound

$$\frac{-\log(1-P(cov.))}{-\log\left(1-\left(\frac{2-\sigma[P]}{2}\right)^{l}\right)} < N$$

Covering bound, cf. [10]

- PAC-learnability of k-DNF problem confirmed for XCS with those bounds!

Optimality Theory on XCS

Motivation

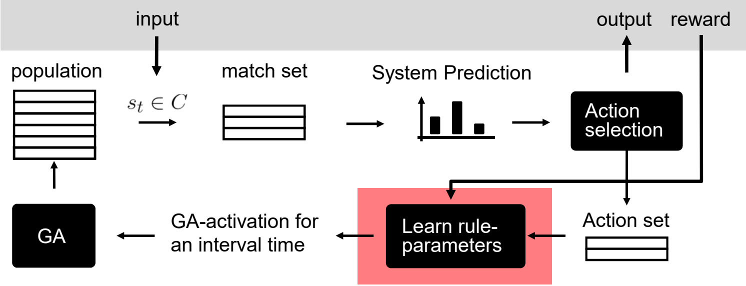
- Latest theoretical studies (Very few)
  - Shift to provide practical insights from hypothetical insights
    - Remove impractical assumptions
      - E.g. infinite iteration, Wilson's generalization hypothesis
    - Capture the optimality of the XCS framework to maximize the performance
  - Theoretical analysis for the "whole" behavior of XCS
    - How rule-learning affect rule-evolution?
    - Is there any "sweet spot" to achieve both the optimality of rule-learning and evolution?
    - A lot of things that we should reveal a complexity of evolutionary rule-based learning

- Which optimality we have known so far?
  - Optimality on Rule-learning (theoretically-validated) [31, 69]
  - Optimality on Rule-evolution (hypothetical) [30]
  - Dilemma between Rule-learning and Rule-evolution (theoretically-validated) [68]

Optimality Theory on XCS

Optimality on Rule-learning

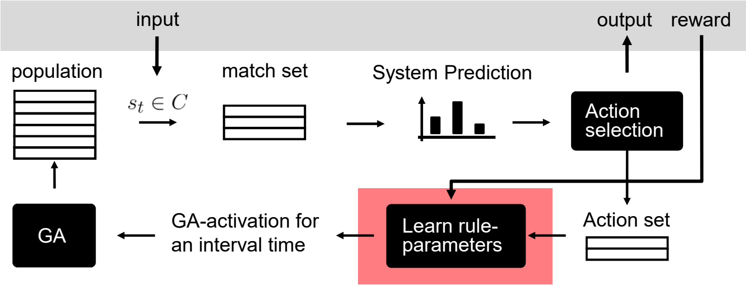
- Learning capacity: To estimate the true-worth of rules
  - On classification tasks: confirmed [31, 69]
  - XCS Learning Theory enables XCS to identify accurate rules in as few training instances as possible



Optimality Theory on XCS

Optimality on Rule-evolution

- Search capacity: To generate accurate rules
    - Non-deterministic, so hard to describe the optimality
    - Can we still say deterministic optimality to search capacity?



Optimality on Rule-learning
Overview

- Unconfirmed main capacities of XCS
  - To generate accurate rules
  - To estimate the true-worth of rules ← focus

- Learning optimality theory [31, 69]
  - **Optimality**: theoretically guarantee that XCS correctly distinguish accurate rules from inaccurate rules with the minimum training
  - **Benefit1**: guideline to set the optimum parameter values of the XCS learning parameters
  - **Benefit2:** you can get optimality on your LCS if your LCS employs the same learning scheme as in XCS
  - **Restriction**: applicable only to classification problems with binary reward scheme (so far)

Optimality on Rule-learning
Brief description 1/4

- Easy step to get optimality on the XCS learning scheme 1/3

- Definition
  - A classification task with binary reward scheme
    - Correct class: a positive reward r_{max}
    - Incorrect class: a negative reward r_{min}
  - Accurate rules boundary:
    - Accurate rules: 100% classification accuracy $P_C{}^* = 1.0$
    - Inaccurate rules: < 100% classification accuracy $P_C{}' < P_C{}^*$
  - In fact, we can control the quality of inaccurate rule with $P_{C'_{\max}}$
    - Set $P_{C'_{\max}}$ to a user's defined value
    - Accurate rules (redefined): having $P_{C'_{\max}}$ % - 100% classification accuracy
    - Inaccurate rules: having <= $P_{C'_{\max}}$ % classification accuracy

Optimality on Rule-learning

Brief description 2/4

- Easy step to get optimality on the XCS learning scheme 1/3

- Goal
  - Guarantee to identify reliably accurate rules correctly
  - Is there any solutions of θ_{sub} , ϵ_0 to satisfy our conditions?

$$n > \theta_{sub}, \underbrace{\epsilon_n}_{\text{controlled by learning rate } \beta} < \epsilon_0 \Rightarrow n^*, n' > \theta_{sub}, \epsilon_{n^*} < \epsilon_{n'}$$

of accurate rules

of inaccurate rules

  - We will answer the following questions

θ_{sub} : *How many times should a rule be updated to be considered for accurate?*

β : *How much rate is adequate to update rules?*

ϵ_0 : *How small a prediction error accurate rules must have?*

Optimality on Rule-learning
Brief description 3/4

- Easy step to get optimality on the XCS learning scheme 1/2

- Step 1
  - Define the quality of accurate rules with $P_{C}{}'_{\max}$

- Step 2
  - Calculate the minimum learning iteration given by

$$\theta_{sub} = \min\{n \in \mathbb{N} \mid 1/(1 - P_{C}{}'_{\max}) \le n\} - 1.$$

- Step 3
  - Find solution β (leaning rate) of the boundary condition by Newton's method:

$$\max \hat{\epsilon}_{\theta_{sub}}(P_C{}^*) = \min \hat{\epsilon}_{\theta_{sub}}(P_C{}').$$

$$\max \hat{\epsilon}_{n^*}(P_C{}^*) = r_{\max}(1-\beta)^{n^*} + r_{\max} n^* \beta (1-\beta)^{n^*-1}.$$

$$\min \hat{\epsilon}_{n'}(P_C{}') = 2 r_{\max}(P_{C}{}'_{\max} - P_{C}{}'^{2}_{\max}) \left| 1 - (1-\beta)^{n'} \right| - r_{\max} n' \beta (1-\beta)^{n'-1} (1 - 2P_{C}{}'_{\max})(P_{C}{}'_{\max} - 1).$$

Optimality on Rule-learning
Brief description 4/4

- Easy step to get optimality on the XCS learning scheme 2/2

- Step 4
  - Set error tolerance ϵ_0 with the solution β as:

$$\epsilon_0 = \max \hat{\epsilon}_{\theta_{sub}}(P_C{}^*)$$

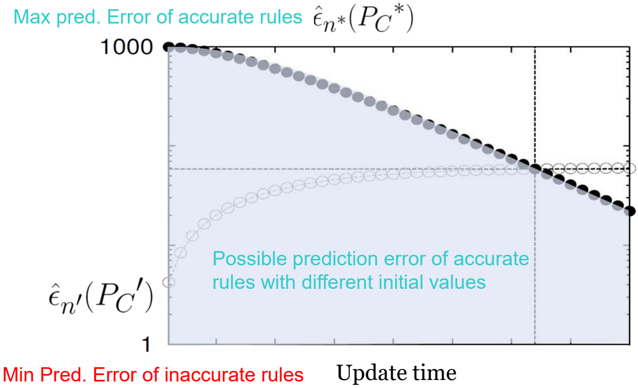
- That's all
  - Determined θ_{sub}, β, ϵ_0 are their optimum values to achieve the optimality on the XCS learning scheme

You can download an open source "theoretically-optimized XCS" at

http://www.nkt.ynu.ac.jp/en/download/

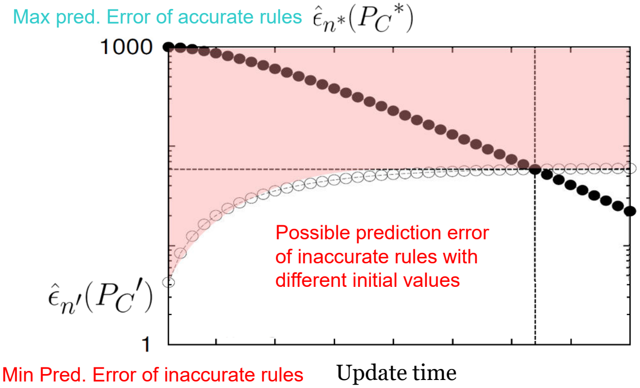
Optimality on Rule-learning

Graphical conclusion 1/3



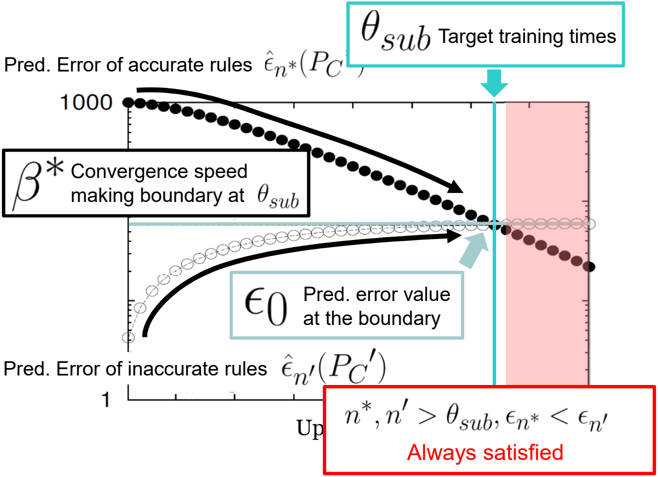
Optimality on Rule-learning

Graphical conclusion 2/3



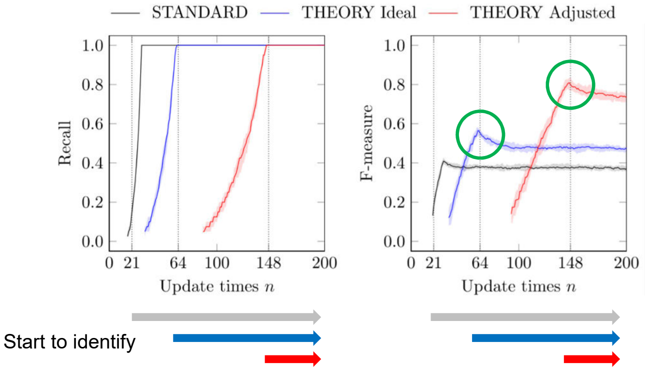
Optimality on Rule-learning

Graphical conclusion 3/3



Optimality on Rule-learning
Impact 1/3

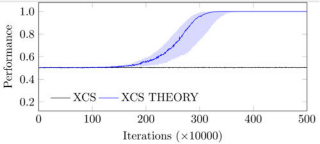
- The optimal parameter settings successfully captures the maximum F-measure score



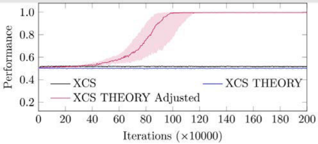
Optimality on Rule-learning
Impact 2/3

- Benchmarks

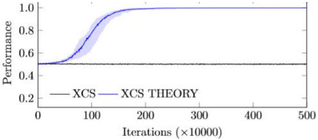
135-MUX: 4.4 E+40 possible inputs
2.6 E+64 possible rules



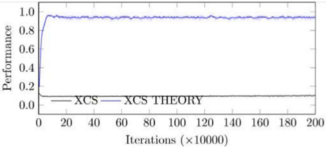
37-MUX: Noisy problem
Optimum = 90% cla. acc.



37-RMUX: real-valued



3x4-CMUX: multi-class problem



Optimality on Rule-learning
Impact 3/3

- Real-world data classification

Dataset	$\lvert D \rvert$	L	#numerical	#categorical	$\lvert C \rvert$	XCS (XCS MAM)	XCS Theory Ideal	XCS Theory Adj. ($P_\alpha = 0.1$)
annealing	898	38	9	29	6	0.847	0.858	0.865
audiology	226	69	0	69	24	0.671	0.718	0.694
australian credit approval	690	14	6	8	2	0.851	0.865	0.862
balance scale	625	4	4	0	3	0.761	0.800	0.808
breast cancer wisconsin	699	9	9	0	2	0.935	**0.958**$^{+}$	0.956
breast cancer wisconsin (diagnostic)	569	30	30	0	2	0.945	0.955	0.945
cardiotocography	2126	21	20	1	10	0.567	**0.647**$^{+}$	**0.710**$^{+}$
congressional voting records	435	16	0	16	2	0.948	0.951	0.955
contraceptive method choice	1473	9	2	7	3	0.481	**0.515**$^{+}$	0.506
dermatology	366	34	33	1	6	0.964	0.969	0.978
ecoli	336	7	3	4	8	0.721	0.739	0.767
glass identification	214	9	9	0	6	0.651	0.652	0.686
heart disease (cleveland)	303	13	6	7	5	0.573	0.542	0.570
heart disease (hungarian)	294	13	6	7	5	0.634	0.656	0.670
hepatitis	155	19	6	13	2	0.791	0.819	0.785
image segmentation	2310	19	19	0	7	0.880	**0.929**$^{+}$	**0.939**$^{+}$
iris	150	4	4	0	3	0.921	0.921	0.893
labor relations	57	16	8	8	2	0.820	0.825	0.735
libras movement	360	90	90	0	15	0.657	0.603	0.683
liver disorders (bupa)	345	6	6	0	2	0.582	0.618	0.606
mushroom	8124	22	0	22	2	1.000	0.999	1.000
primary tumor	339	17	0	17	21	0.289	0.271	**0.404**$^{+}$
sonar	208	60	60	0	2	0.853	0.802	0.823
soybean	683	35	0	35	19	0.588	**0.495**$^{-}$	**0.694**$^{+}$
teaching assistant evaluation	151	5	1	4	3	0.624	0.652	0.645
thyroid disease (sick)	3772	29	7	22	2	0.939	0.939	0.939
vehicle silhouettes	846	18	18	0	4	0.740	0.733	0.763
wine	178	13	13	0	3	0.965	0.940	0.959
yeast	1484	8	7	1	10	0.367	**0.411**$^{+}$	0.385
zoo	101	16	1	15	7	0.879	0.902	0.933
					Average rank:	2.45	1.92	1.63

Optimality on Rule-evolution

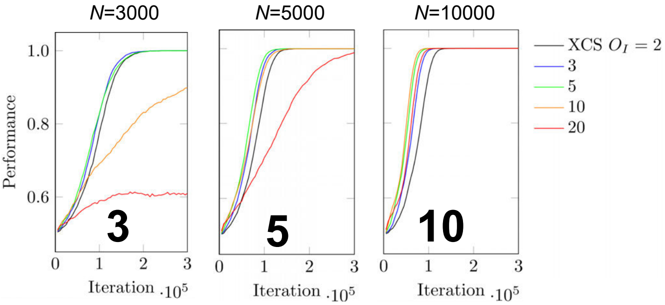
Motivation

- Unconfirmed main capacities of XCS
  - To generate accurate rules ← focus
  - To identify accurate rules

- Optimality on Rule-evolution
  - Hard to guarantee that XCS evolutionary generates accurate rules
  - Instead, we here consider the maximize a probability to generate accurate rules
  - How?
- Optimality hypothesis [30]
  - XCS employs a steady-state GA: generates ONLY two offspring rules for each generation
  - Very inefficient

Optimality on Rule-evolution
Difficulty

- Difficulty to determine the optimal number of generated offspring
  - The problematic cover-delete cycle occurs when increasing the number of generated offspring
- How can XCS safely increase offspring while preventing the cover-delete cycle?



Optimality on Rule-evolution

Optimality hypothesis

- Optimality hypothesis [30]
  - XCS employs a steady-state GA: generates ONLY two offspring rules for each generation
  - Hypothesis:
    - "*A probability to generate accurate rules can be maximized when maximizing the number of offspring rules. Then, the maximum number of offspring rules can be equal to the number of inaccurate rules exist in the current population.*"

- This suggests the optimal number of generated offspring can be dynamically changed and it is corresponding to the number of inaccurate rules existed in the population

- How to safely increase the number of generated solutions?
- → How to safely delete unnecessary rules

Optimality on Rule-evolution
Algorithm

- Very easy to implement this optimality hypothesis

- Step 1
  - Calculate and set the optimal parameter setting derived from the learning optimality theory

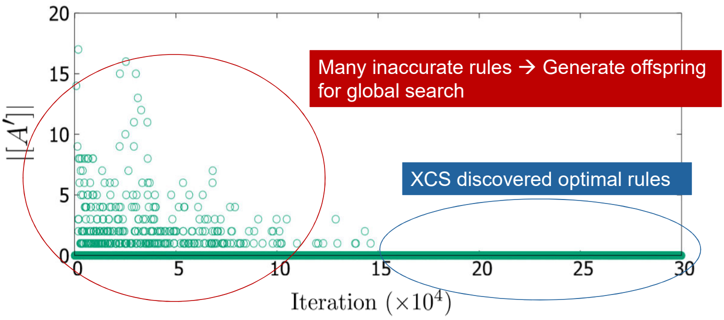
- Step 2
  - Identify the inaccurate rules with the XCS learning scheme

- Step 3
  - Replace inaccurate rules with newly-generated offspring rules

- That's all

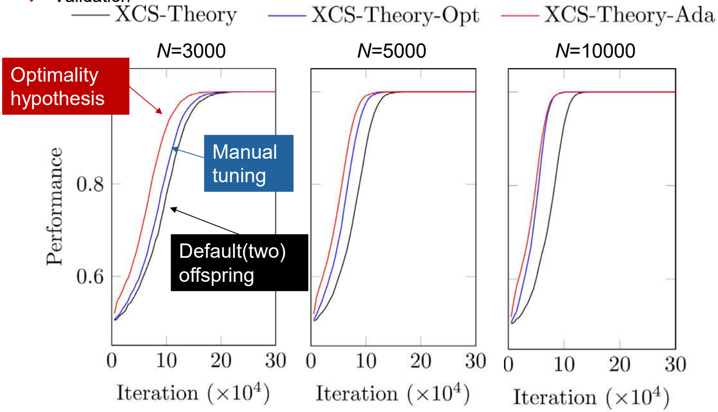
Optimality on Rule-evolution
Self-adaptation

- Optimality hypothesis works as self-adaption of the number of offspring



Optimality on Rule-evolution
Impact 1/2

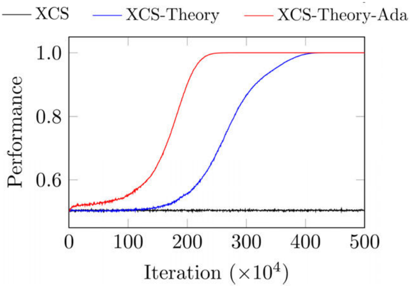
- Validation



Optimality on Rule-evolution
Impact 2/2

❖ Impact of Optimality hypothesis

135-MUX: 4.4 E+40 possible inputs
2.6 E+64 possible rules



Optimality Theory on XCS

Summary

- Seeking of the optimality of the XCS framework
  - Optimality on Rule-learning:
    - Partially done in 2017 & 2020 (for classification problems)
    - Not yet for multiple reward scheme and for reinforcement learning task

    - Provides a reasonable guideline to set the XCS learning parameters
    - Easy to use (Get optimality in your XCS-based systems)
    - Theoretically-reliable extensions, e.g. self-adaptation of learning parameters [70]
  - Optimality on Rule-evolution:
    - Yet restricted in hypothetical insight
    - Hypothetical insight still work
- Join us
  - Theoretical works gradually get attention….
    - Potential to drastically improve the LCS performance
    - I.e. bottom-up of evolutionary symbolic approach like LCS
    - One of the “well theoretically-studied” evolutionary machine learning variants
  - A lot of things that we should reveal
    - For multi-step problems
    - Wilson’s generalization hypothesis…

Course Agenda

- Introduction
  - ✓ A Brief Definition
  - ✓ Why LCS?
  - ✓ Looking Back: LCS History
- Michigan-style Learning Classifier Systems
  - ✓ Building Blocks of LCS
  - ✓ Putting it together: A generic LCS
  - ✓ Bridging the Gap: Approaching XCS
- XCS Theory in a Nutshell
  - ✓ An Overview of Formal Theory Behind LCS
  - ✓ Learning Optimality Theory
- **Modern Systems**
  - XCSF: Piece-wise Online Function Approximation
  - ExSTraCS: Large-scale Supervised Classification
- Summary & Conclusions
  - A Different Perspective
  - Why LCS?
  - Resources & Current Research

Modern Systems
XCSF: Piece-wise Online Function Approximation

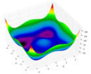
- XCS for function approximation introduced by Wilson in 2002 [64]
  - Supervised learning → Actions become obsolete; only dummy action a_d
  - Online learning → Adapt model instance per instance
  - Local learning → Classifiers partition the input space; divide-and-conquer
  - Evolutionary Learning → Steady-state niche GA optimizes input space coverage

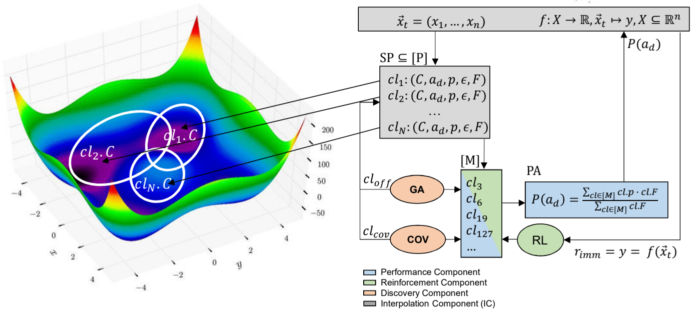
- **Alternative view:** Evolutionary Ensemble Learner
  - XCS' algorithmic structure as a general online ensemble learning framework
  - Classifiers as members of that ensemble
  - No Boosting, no Bagging, more like Stacking
  - Allows hybrid ensemble (cf. [26])

Modern Systems

XCSF: Piece-wise Online Function Approximation

Schematic



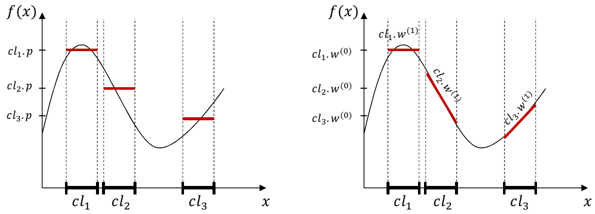


Modern Systems

XCSF: Innovations to preceding XCS(R) (1/2)

- **Development of Classifier Prediction**
  - 90's: Wilson introduced ZCS and XCS as reinforcement learning algorithms
    - Classifiers cl advocate specific action $cl.a \in A$ for certain subset of states $\{\vec{x}_i\} \subseteq X$
    - Prediction attribute $cl.p$ was defined to estimate the expected reward $\mathbb{E}[r|\vec{x}, a]$.
  - 2000: XCS recognized to be well applicable to supervised learning tasks (classification).
  - since 2001: Not surprisingly, it was then also used to approximate functions (regression).
    - Prediction $cl.p$ was used as XCS' output
    - Eventually, modeled as function $f(x) = \vec{w}^T\vec{x} + w_0$ of the current input $\vec{x} \in X$

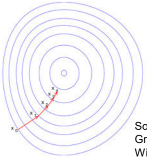
- **Intuition**



Modern Systems

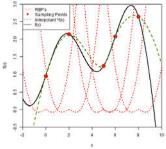
XCSF: Innovations to preceding XCSR (2/2)

- **Competent update procedures** (cf. Lanzi et al. [24])
  - Linear Least Square
  - Kalman Filter
  - Gain Adaptation
  - Recursive Least Square

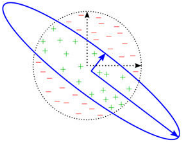


Source:
Gradient descent,
Wikipedia

- **Various predictors**
  - Polynomial approximation [25]
  - Evolution Strategy [48]
  - Neural Network [23]
  - Support Vector Regression [29]
  - RBF-Interpolation [42]



- **Guided Mutation** [37]
  - Inspired by Covariance Matrix Adaptation
  - Store weights for matching samples
  - Assign weight < 1 for instances with high error (and vice versa)
  - Guide mutation towards positively weighted instances



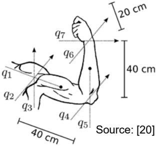
Source: [37]

Modern Systems

XCSF: Applications

❖ Robot Kinematics

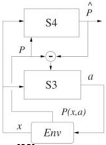
- Filtering of sensory information [20]
- Locally linear forward kinematics [38]



Source: [20]

❖ Continuous action spaces [63]

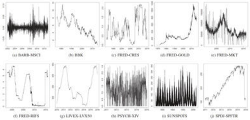
- Hierarchical XCSF architectures
- e.g., Continuous Actor-Critic approach



Source: [63]

❖ Stacking Approach for Ensemble Forecasting [36]

- Use of hybrid forecasting techniques (ARIMA, Exp. Smooting, etc.)
- Locally learning the weights for combination of those
- Applied to different time series



Source: [36]

Modern Systems

ExSTraCS: Large-scale Supervised Classification

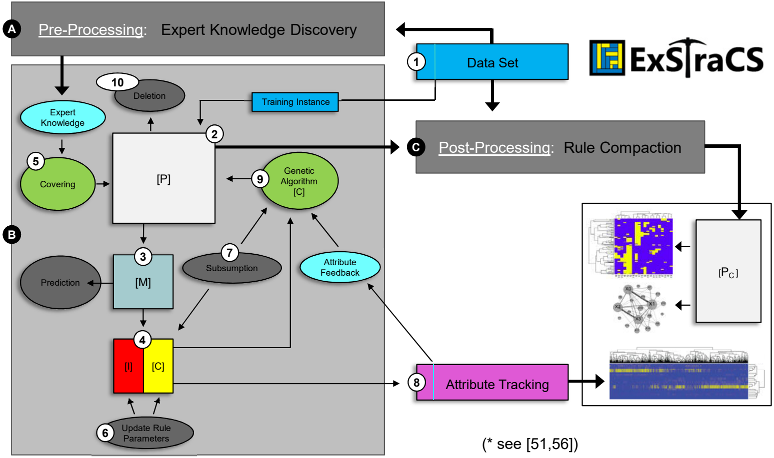
- First introduced by Urbanowicz and Moore in 2014 [56]
- Conceived to tackle large-scale, complex classification problems
- Equipped with mechanisms for post-hoc Knowledge Discovery
- Proved very successful in large multiplexer problems (135-bit!)
- Focus on LCS scalability in terms of:
  - Increasing number of training instances (big data)
  - Increase in problem dimensionality (relevant features)
  - Increase in total number of features (curse of dimensionality)

- Open Source project (Python): https://github.com/ryanurbs/ExSTraCS_2.0

- Visit hands-on session at **IWLCS**@GECCO!

Modern Systems

ExSTraCS: Overview



(* see [51,56])

* Adapted from Urbanowicz's previous tutorials

Modern Systems

A Pre-Processing:

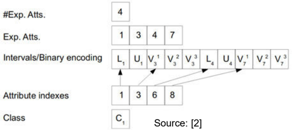
ExSTraCS: Adaptive Data Management (ADM)

- Automatically calculate training data statistics:
  - Number of attributes
  - Number of instances
  - Location of endpoint (class)

- Automatic shuffling to prevent bias

- Determines data characteristics:
  - Location of categorical attributes
  - Location of continuous attributes
  - Determines min and max ranges
  - Counts distinct values for each attribute within the training data

ALKR-style encoding

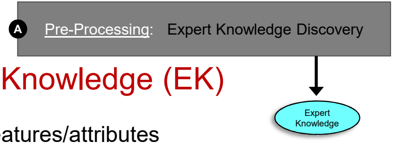


Source: [2]

- Automatic selection of Rule-Specificity limit (RSL)

Modern Systems

ExSTraCS: Using Expert Knowledge (EK)

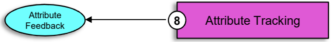


- Expert provides weights to the features/attributes
- Weights determine `predictive value’
- Weights guide covering mechanism and GA

- Weights can be provided manually by expert user, or…
- … automatically by utilizing Relief-based attribute weighting
  - ReliefF, SURF, SURF*, MultiSURF
  - New to ExSTraCS 2.0 → Tuned-ReliefF (TuRF)

- Introduces sort of automated feature selection
- But: without actual removal for knowledge discovery purposes!

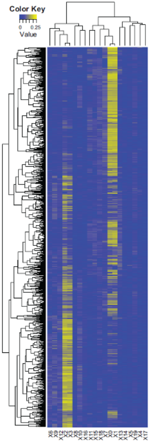
* see [51,55] for more details



Modern Systems

ExSTraCS: Attribute Tracking und Feedback (AT&F)

- An extension to the LCS algorithm that allows for the explicit characterization of heterogeneity, and allows for the identification of heterogeneous subject groups.
- Akin to long-term memory. Experiential knowledge stored separately from the rule population that is never lost.
- Relies on learning that is both incremental and supervised.
- Stored knowledge may be fed back into LCS during learning.

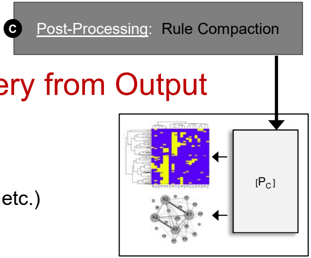


* Adapted from Urbanowicz's previous tutorials

* see [54,57] for more details

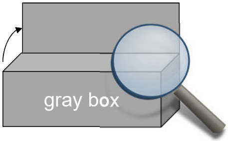
Modern Systems

ExSTraCS: Knowledge Discovery from Output



- ❖ Outputs up to 5 distinct output files
  a) Final population of learned rules
  b) Population metrics (train/test accuracy, etc.)
  c) Attribute co-occurrence in final rules
  d) Attribute tracking scores per instance
  e) Summary of predictions for testing data, including votes (for further use)

- ❖ Facilitate algorithm transparency and interpretability!



* see [53,57] for more details

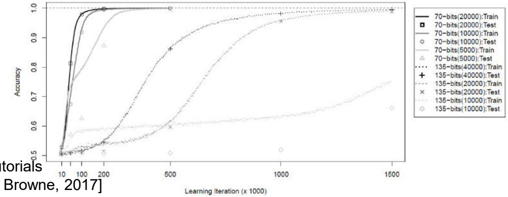
Modern Systems
ExSTraCS: Solving the 135-Multiplexer

x	Address Bits	Order of Interaction	Heterogeneous Combinations	Unique Instances	Optimal Rules [O]
6-bit	2	3	4	64	8
11-bit	3	4	8	2048	16
20-bit	4	5	16	1.05×10^{6}	32
37-bit	5	6	32	1.37×10^{11}	64
70-bit	6	7	64	1.18×10^{21}	128
135-bit	7	8	128	4.36×10^{40}	256

- TO SOLVE: 135-bit Multiplexer
  - All 135 features are predictive in at least some subset of the dataset.
  - Non-RBML approaches would need to include all 135 attributes together in a single model properly capturing underlying epistasis and heterogeneity.

- Few ML algorithms can make the claim that they can solve even the 6 or 11-bit multiplexer problems, let alone the 135-bit multiplexer.

ExSTraCS



* Adapted from Urbanowicz's previous tutorials
* Images adapted from [Urbanowicz and Browne, 2017]

Course Agenda

- Introduction
  - ✓ A Brief Definition
  - ✓ Why LCS?
  - ✓ Looking Back: LCS History
- Michigan-style Learning Classifier Systems
  - ✓ Building Blocks of LCS
  - ✓ Putting it together: A generic LCS
  - ✓ Bridging the Gap: Approaching XCS
- XCS Theory in a Nutshell
  - ✓ An Overview of Formal Theory Behind LCS
  - ✓ Learning Optimality Theory
- Modern Systems
  - ✓ XCSF: Piece-wise Online Function Approximation
  - ✓ ExSTraCS: Large-scale Supervised Classification
- **Summary & Conclusions**
  - A Different Perspective
  - Why LCS?
  - Resources & Current Research

Summary & Conclusions
A Different (ML-centric) Perspective on LCS

- Reconsider M-style LCS as an online ensemble learner
- Rules = ensemble members
- Each rule constitutes a local model / hypothesis
- Rules are experts of different problem niches→ mixture of experts
- `Goodness' of each `expert' determined instance-by-instance without necessity to remember → one-pass (online) learning
- Modularity (recall building block intuition) allows for stacking
  - Different models for local prediction (ANN, RBF, polynomials) and fitness-weighted combination = stacked generalization
- Learning Classifier System not only about classification (alone)
  - XCSF: Function Approximation = Regression
  - XCS(R): Sequential Decision Making = Reinforcement Learning
  - XCSC: Clustering = Unsupervised Learning
- XCSF → similarities to Locally Weighted Projection Regression
- XCS(R) → generalizing Q-learner

Summary & Conclusions

So, again: Why LCS? (ex post)

- Flexibility (RL, SL) and modularity (building blocks)
- Interpretability by design (condition-action rules)
- Follow divide and conquer principle (mixture of experts)
- Capture complex associations (epistasis, heterogeneity)
- Evolution as central component allows adaptation to change (concept drift)
- Overarching framework for general ML techniques
  - LCS and Deep Learning do not mutually exclude!
  - E.g., put DNNs to locally model a policy
- And (again) finally…
  - they are simply cool ;-)

Summary & Conclusions
Recent Research Directions (excerpt)

- Visual and statistical knowledge discovery from LCS rule sets (Urbanowicz et al. [57])
- Theoretical hyperparameter derivation (Nakata et al. [30,31])
- Hierarchical LCS and multi-domain learning (Liu, Browne, Xue [28])
- Absumption for Classifier Specialization (Liu, Browne, Xue [65])
- Lexicase Selection for Supervised LCS (Aenugu and Spector [67])
- LCS with active learning (Stein et al. [41])
- Experience Replay & Interpolation in XCS (Stein et al. [66,40,42,43])
- Algebraic formalization of LCS (Pätzel and Hähner [32])
- …
- → nearly all of them regularly attend GECCO!

Thanks!

You feel triggered and want to learn for more?

Don’t miss the annual

International Workshop on
Learning Classifier Systems (IWLCS)

at GECCO

Acknowledgements

Thanks to Ryan J. Urbanowicz for the permission to reuse parts of his previous tutorials on LCS.

Resources

- Additional Information:
  - Keep up to date with the latest LCS research
  - Get in contact with an LCS researcher
  - Contribute to the LCS community research and discussions.

- GBML Central - http://gbml.org/

- LCS Researcher Webpages:
  - Bacardit, Jaume - http://homepages.cs.ncl.ac.uk/jaume.bacardit/
  - Browne, Will - http://ecs.victoria.ac.nz/Main/WillBrowne
  - Bull, Larry - http://www.cems.uwe.ac.uk/~lbull/
  - Holmes, John - https://www.med.upenn.edu/apps/faculty/index.php/g5455356/p19936
  - Kovacs, Tim - http://www.cs.bris.ac.uk/home/kovacs/
  - Lanzi, Pier Luca - http://www.pierlucalanzi.net/
  - Nakata, Masaya - http://www.nkt.ynu.ac.jp/en/people/
  - Stein, Anthony - https://ki-agrartechnik.uni-hohenheim.de/anthony-stein
  - Urbanowicz, Ryan - http://www.ryanurbanowicz.com/
  - Wilson, Stewart - https://www.eskimo.com/~wilson/

- International Workshop Learning Classifier Systems (IWLCS) - held annually at GECCO

- Mailing List:: Yahoo Group: lcs-and-gbml[at]yahoogroups.com

* Adapted from Urbanowicz's previous tutorials

Resources: Available Software

- Scitkit-compatible LCS (scikit-eLCS) – in Python.
  - https://github.com/robertfrankzhang/Scikit-eLCS
  - A sklearn-compatible Python implementation of eLCS, a supervised learning variant of the Learning Classifier System, based off of UCS.
- Educational LCS (eLCS) – in Python.
  - https://github.com/ryanurbs/eLCS
  - Simple Michigan-style LCS for learning how they work and how they are implemented.
  - Code intended to be paired with first LCS introductory textbook by Urbanowicz/Browne.
- ExSTraCS 2.0 – Extended Supervised Learning LCS – in Python
  - https://github.com/ryanurbs/ExSTraCS_2.0
  - For prediction, classification, data mining, knowledge discovery in complex, noisy, epistatic, or heterogeneous problems.
- BioHEL – Bioinformatics-oriented Hierarchical Evolutionary Learning – in C++
  - http://ico2s.org/software/biohel.html
  - GAssist also available through this link.
- XCSLib (XCS and XCSF) (by Lanzi in C++)
  - http://xcslib.sourceforge.net/
- XCSF with function approximation visualization – in Java
  - Martin Butz Chair website

* Adapted from Urbanowicz's previous tutorials

Resources: LCS Review Papers & Books

Selected Review Papers:

- Pätzel, David, Stein, Anthony, and Hähner, Jörg. "A Survey on Formal Theoretical Advances Regarding XCS." *Proc. of GECCO '19 Companion, July 2019, 1295-1302*
- Bull, Larry. "A brief history of learning classifier systems: from CS-1 to XCS and its variants." *Evolutionary Intelligence* (2015): 1-16.
- Bacardit, Jaume, and Xavier Llorà. "Large-scale data mining using genetics-based machine learning." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 3.1 (2013): 37-61.
- Urbanowicz, Ryan J., and Jason H. Moore. "Learning classifier systems: a complete introduction, review, and roadmap." *Journal of Artificial Evolution and Applications* 2009 (2009): 1.
- Sigaud, Olivier, and Stewart W. Wilson. "Learning classifier systems: a survey." *Soft Computing* 11.11 (2007): 1065-1078.
- Holland, John H., et al. "What is a learning classifier system?." *Learning Classifier Systems*. Springer Berlin Heidelberg, 2000. 3-32.
- Lanzi, Pier Luca, and Rick L. Riolo. "A roadmap to the last decade of learning classifier system research (from 1989 to 1999)." *Learning Classifier Systems*. Springer Berlin Heidelberg, 2000. 33-61.

Books:

- Drugowitsch, J., (2008) Design and Analysis of Learning Classifier Systems: A Probabilistic Approach. Springer-Verlag.
- Bull, L., Bernado-Mansilla, E., Holmes, J. (Eds.) (2008) Learning Classifier Systems in Data Mining. Springer
- Butz, M (2006) Rule-based evolutionary online learning systems: A principled approach to LCS analysis and design. Studies in Fuzziness and Soft Computing Series, Springer.
- Bull, L., Kovacs, T. (Eds.) (2005) Foundations of learning classifier systems. Springer.
- Kovacs, T. (2004) Strength or accuracy: Credit assignment in learning classifier systems. Springer.
- Butz, M. (2002) Anticipatory learning classifier systems. Kluwer Academic Publishers.
- Lanzi, P.L., Stolzmann, W., Wilson, S., (Eds.) (2000). Learning classifier systems: From foundations to applications (LNAI 1813). Springer.
- Holland, J. H. (1975). Adaptation in natural and artificial systems. University of Michigan Press.

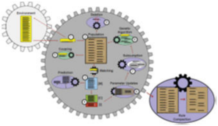
* Adapted from Urbanowicz's previous tutorials

Resources: Most recent

- Textbook: '*Introduction to Learning Classifier Systems*'
  Springer, 2017 (Urbanowicz & Brown, 2017)

- LCS Introductory Chapter: 'Reaction Learning', Chapter 7.1 in book: '*Organic Computing – Technical Systems for Survival in the Real World'*, Birkhäuser, 2017 (Stein, 2017)

- YouTube video on LCS:
  - Learning Classifier Systems in a Nutshell
  - Animated, narrated explanation of basic LCS concepts.
  - https://www.youtube.com/watch?v=CRge_cZ2cJc



- LCS and Rule-Based Machine Learning Wikipedia Pages – recently updated and revised.
  (https://en.wikipedia.org/wiki/Learning_classifier_system)

* Adapted from Urbanowicz's previous tutorials

References
Figures

- Figure sources: All figures that have not been created by the author or indicated otherwise are free to use and taken from pixabay.com licensed according to the Pixaybay License

References (1/6)

(1) Bacardit, Jaume, et al. "Speeding-up Pittsburgh learning classifier systems: Modeling time and accuracy." *Parallel Problem Solving from Nature-PPSN VIII*. Springer Berlin Heidelberg, 2004.

(2) Bacardit, Jaume, and Natalio Krasnogor. "A mixed discrete-continuous attribute list representation for large scale classification domains." *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009.

(3) Bacardit, Jaume, and Xavier Llorà. "Large-scale data mining using genetics-based machine learning." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 3.1 (2013): 37-61.

(4) Bernadó-Mansilla, Ester, and Josep M. Garrell-Guiu. "Accuracy-based learning classifier systems: models, analysis and applications to classification tasks."*Evolutionary Computation* 11.3 (2003): 209-238.

(5) Booker, Lashon Bernard. "Intelligent behavior as an adaptation to the task environment, University of Michigan." *Ann Arbor, MI* (1982).

(6) Bull, Larry. "A simple accuracy-based learning classifier system." *Learning Classifier Systems Group Technical Report UWELCSG03-005, University of the West of England, Bristol, UK* (2003).

(7) Bull, Larry, and O'Hara, Toby. "Accuracy-based neuro and neuro-fuzzy classifier systems.", *GECCO 2002, 905-911, Morgan Kaufmann, 2002*.

(8) Butz, M.; Kovacs, T.; Lanzi, P. & Wilson, S., "Toward a Theory of Generalization and Learning in XCS", IEEE Transactions on Evolutionary Computation, 8 , 28-46, 2004

(9) Butz, M.; Lanzi, P. & Wilson, S. "Function Approximation With XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction", IEEE Transactions on Evolutionary Computation, 12 , 355-376, 2008

(10) Butz, M. V., "Rule-based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design", Springer, 2005

(11) Frey, Peter W., and David J. Slate. "Letter recognition using Holland-style adaptive classifiers." *Machine Learning* 6.2 (1991): 161-182.

(12) Goldberg, David E. "E. 1989. Genetic Algorithms in Search, Optimization, and Machine Learning." *Reading: Addison-Wesley* (1990).

References (2/6)

(13) Goldberg, D. E., “Genetic Algorithms as a Computational Theory of Conceptual Design”, Rzevski, G. & Adey, R. A. (Eds.), Applications of Artificial Intelligence in Engineering VI, Springer Netherlands, 1991 , 3-16

(14) Holland, J., and J. Reitman. "Cognitive systems based on adaptive agents.”, *Pattern-directed inference systems* (1978).

(15) Holland, J. “Properties of the Bucket brigade.*” In Proceedings of the 1st International Conference on Genetic Algorithms*, 1-7 (1985)

(16) Holmes, John H. "A genetics-based machine learning approach to knowledge discovery in clinical data." *Proceedings of the AMIA Annual Fall Symposium*. American Medical Informatics Association, 1996.

(17) Holmes, John H., and Jennifer A. Sager. "The EpiXCS workbench: a tool for experimentation and visualization." *Learning Classifier Systems*. Springer Berlin Heidelberg, 2007. 333-344.

(18) Iqbal, Muhammad, Will N. Browne, and Mengjie Zhang. "Extending learning classifier system with cyclic graphs for scalability on complex, large-scale boolean problems." *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013.

(19) Iqbal, M.; Browne, W. N. & Zhang, M., “Reusing Building Blocks of Extracted Knowledge to Solve Complex, Large-Scale Boolean Problems”, IEEE Transactions on Evolutionary Computation, 2014 , 18 , 465-480

(20) Kneissler, J.; Stalph, P. O.; Drugowitsch, J. & Butz, M. V., “Filtering Sensory Information with XCSF: Improving Learning Robustness and Robot Arm Control Performance”, Evolutionary Computation, 2014 , 22, 139-158

(21) Kovacs, Tim. "A comparison of strength and accuracy-based fitness in learning classifier systems." *School of Computer Science, University of Birmingham, Birmingham, UK* (2002).

(22) Kovacs, Tim. "What should a classifier system learn and how should we measure it?." *Soft Computing* 6.3-4 (2002): 171-182.

(23) Lanzi, P. L. & Loiacono, D., “XCSF with Neural Prediction”, IEEE CEC, 2006 , 2270-2276

(24) Lanzi, P. L.; Loiacono, D.; Wilson, S. W. & Goldberg, D. E., “Generalization in the XCSF Classifier System: Analysis, Improvement, and Extension”, Evol. Comput., MIT Press, 2007, 15, 133-168

(25) Lanzi, P. L.; Loiacono, D.; Wilson, S. W. & Goldberg, D. E., “Extending XCSF Beyond Linear Approximation”, GECCO 2005, ACM, 2005, 1827-1834

References (3/6)

(26) Lanzi, P. L.; Loiacono, D. & Zanini, M., “Evolving classifier ensembles with voting predictors”, IEEE CEC 2008, June 1-6, 2008, Hong Kong, China, 2008 , 3760-3767

(27) Lanzi, P. L. & Wilson, S. W., “Using Convex Hulls to Represent Classifier Conditions”, GECCO 2006, ACM, 2006, 1481-1488

(28) Liu, Y.; Xue, B. & Browne, W. N., “Visualisation and Optimisation of Learning Classifier Systems for Multiple Domain Learning”, Simulated Evolution and Learning, Springer International Publishing, 2017, 448-461

(29) Loiacono, D.; Marelli, A. & Lanzi, P. L., “Support vector regression for classifier prediction”, GECCO 2007, 2007, 1806-1813

(30) Nakata, M.; Browne, W. N. & Hamagami, T., “Theoretical adaptation of multiple rule-generation in XCS”, GECCO 2018, Kyoto, Japan, July 15-19, 2018 , 482-489

(31) Nakata, M.; Browne, W. N.; Hamagami, T. & Takadama, K., “Theoretical XCS parameter settings of learning accurate classifiers”, GECCO 2017, Berlin, Germany, July 15-19, 2017, 2017 , 473-480

(32) Pätzel, D. & Hähner, J., “An Algebraic Description of XCS”, GECCO 2018 Companion, ACM, 2018, 1434-1441

(33) Pätzel, D.; Stein, A. & Hähner, J., “A Survey on Formal Theoretical Advances Regarding XCS”, GECCO’19 Companion, ACM, July 2019, 1295-1302

(34) Riolo, Rick L. "Lookahead planning and latent learning in a classifier system."*Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*. MIT Press, 1991.

(35) Smith, Stephen Frederick. "A learning system based on genetic adaptive algorithms.“, Dissertation, University of Pittsburgh, 1980.

(36) Sommer, M.; Stein, A. & Hähner, J., “Local ensemble weighting in the context of time series forecasting using XCSF”, IEEE SSCI, 2016

(37) Stalph, P. O. & Butz, M. V., “Guided Evolution in XCSF”, GECCO 2012, ACM, 2012, 911-918

(38) Stalph, P. O. & Butz, M. V., “Learning local linear Jacobians for flexible and adaptive robot arm control”, GPEM, 2012, 13, 137-157

(39) Stein, A., “Reaction Learning”, Ch. Basic Methods, In book: Organic Computing -- Technical Systems for Survival in the Real World, Müller-Schloer, C. & Tomforde, S. (Eds.), Birkhäuser, 2017, 287-328

References (4/6)

(40) Stein, A.; Eymüller, C.; Rauh, D.; Tomforde, S. & Hähner, J., “Interpolation-based Classifier Generation in XCSF”, IEEE CEC, 2016 , 3990-3998

(41) Stein, A.; Maier, R. & Hähner, J., “Toward Curious Learning Classifier Systems: Combining XCS with Active Learning Concepts”, GECCO 2017 Companion, ACM, 2017, 1349-1356

(42) Stein, A.; Menssen, S. & Hähner, J., “What About Interpolation? A Radial Basis Function Approach to Classifier Prediction Modeling in XCSF”, GECCO 2018, ACM, 2018

(43) Stein, A.; Rauh, D.; Tomforde, S. & Hähner, J., “Interpolation in the eXtended Classifier System: An Architectural Perspective”, Journal of Systems Architecture, 75, 79-94, 2017

(44) Stolzmann, Wolfgang. "An introduction to anticipatory classifier systems."*Learning Classifier Systems*. Springer Berlin Heidelberg, 2000. 175-194.

(45) Stone, Christopher, and Larry Bull. "For real! XCS with continuous-valued inputs." *Evolutionary Computation* 11.3 (2003): 299-336.

(46) Tamee, K.; Bull, L. & Pinngern, O., “Towards Clustering with XCS”, GECCO 2007, ACM, 2007, 1854-1860

(47) Tan, J.; Moore, J. & Urbanowicz, R., “Rapid Rule Compaction Strategies for Global Knowledge Discovery in a Supervised Learning Classifier System”, The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE), 2013, 110-117

(48) Tran, T. H.; Sanza, C. & Duthen, Y., “Evolving prediction weights using evolution strategy”, GECCO 2008, 2009-2016, 2008

(49) Urbanowicz, Ryan J., and Jason H. Moore. "Learning classifier systems: a complete introduction, review, and roadmap." *Journal of Artificial Evolution and Applications* 2009 (2009): 1.

(50) Urbanowicz, Ryan J., and Will Browne. “An Introduction to Learning Classifier Systems”. Springer, 2017

(51) Urbanowicz, Ryan J., and Jason H. Moore. "ExSTraCS 2.0: description and evaluation of a scalable learning classifier system." *Evolutionary Intelligence*(2015): 1-28.

(52) Urbanowicz, Ryan J., and Jason H. Moore. "The application of michigan-style learning classifier systems to address genetic heterogeneity and epistasis in association studies." *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 2010.

References (5/6)

(53) Urbanowicz, Ryan J., Ambrose Granizo-Mackenzie, and Jason H. Moore. "An analysis pipeline with statistical and visualization-guided knowledge discovery for michigan-style learning classifier systems." *Computational Intelligence Magazine, IEEE* 7.4 (2012): 35-45.

(54) Urbanowicz, Ryan, Ambrose Granizo-Mackenzie, and Jason Moore. "Instance-linked attribute tracking and feedback for michigan-style supervised learning classifier systems." *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, 2012.

(55) Urbanowicz, Ryan J., Delaney Granizo-Mackenzie, and Jason H. Moore. "Using expert knowledge to guide covering and mutation in a michigan style learning classifier system to detect epistasis and heterogeneity." *PPSN XII*. Springer Berlin Heidelberg, 2012. 266-275.

(56) Urbanowicz, R. J.; Bertasius, G. & Moore, J. H., “An Extended Michigan-Style Learning Classifier System for Flexible Supervised Learning, Classification, and Data Mining”, PPSN XIII, Springer International Publishing, 2014, 211-221

(57) Urbanowicz, R. J.; Lo, C.; Holmes, J. H. & Moore, J. H., “Attribute Tracking: Strategies Towards Improved Detection and Characterization of Complex Associations”, GECCO 2018, ACM, 2018, 553-560

(58) Urbanowicz, R. J. & Browne, W. N., “Introduction to Learning Classifier Systems”, Springer Publishing Company, 2017

(59) Wilson, Stewart W. "ZCS: A zeroth level classifier system." *Evolutionary computation* 2.1 (1994): 1-18.

(60) Wilson, Stewart W. "Classifier fitness based on accuracy." *Evolutionary computation* 3.2 (1995): 149-175.

(61) Wilson, Stewart W. "Get real! XCS with continuous-valued inputs." *Learning Classifier Systems*. Springer Berlin Heidelberg, 2000. 209-219.

(62) Wilson, Stewart W. "Classifiers that approximate functions." *Natural Computing*1.2-3 (2002): 211-234.

(63) Wilson, S., “Three Architectures for Continuous Action”, Learning Classifier Systems, Springer Berlin Heidelberg, 2007 , 4399 , 239-257

(64) Wilson, S. W., “Classifiers that Approximate Functions”, Natural Computing, Kluwer Academic Publishers, 2002, 1, 211-234

References (6/6) – new since GECCO 2019

(65) Liu, Y., Browne, W. N., Xue, B. “Absumption to Complement Subsumption in Learning Classifier Systems”, Proc. of GECCO ‘19, ACM, July 2019, 410–418

(66) Stein, A., Maier, R., Rosenbauer, L., Hähner, J. “XCS Classifier System with Experience Replay”, Proc. of GECCO ‘20, ACM, 2020, accepted for publication

(67) Aenugu, S., Spector, L. “Lexicase Selection in Learning Classifier Systems”, Proc. of GECCO ‘19, ACM, July 2019, 356–364

(68) Nakata, M., Browne. W. “How XCS Can Prevent Misdistinguishing Rule Accuracy: A Preliminary Study”, Proc. of GECCO ’19 Companion, ACM, July 2019, 183-184

(69) Nakata, M., Browne. W. “Learning Optimality Theory for Accuracy-based Learning Classifier Systems”, IEEE Transactions on Evolutionary Computation, 2020, accepted for publication

(70) Horiuchi, M., Nakata, M. “Self-adaptation of XCS learning parameters based on Learning theory”, Proc. of GECCO ‘20, ACM, 2020, accepted for publication