Learning Classifier Systems: Introducing the User-friendly Textbook

Will Browne[1] & Ryan Urbanowicz[2]
1. Evolutionary Computation Research Group, Victoria University of Wellington, NZ
Will.Browne@vuw.ac.nz
2. Dartmouth College, USA
RyanUrbanowicz@gmail.com

http://www.sigevo.org/gecco-2013/

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

Guides

- **Will N. Browne's** main area of research is Artificial Cognitive Systems. He has served as Co-track chair for Genetics-Based Machine Learning in GECCO 2011 and 2012 [plus organizing committee of the International Workshop on Learning Classifier Systems (IWLCS) from 2009-2010]. Editor in chief for the Australasian Conference on Robotics and Automation 2012 and organized the Cognitive Robotics Intelligence and Control COGRIC, EPSRC (UK)/NSF (USA)
- **Ryan Urbanowicz** holds a Ph.D in genetics from Dartmouth College, and both a M.Eng. and B.Eng. in agricultural and biological engineering from Cornell University. His current research focuses on the development of machine learning strategies for feature selection, modeling, classification, and data mining in studies of common complex human disease. He served on the IWLCS organizing committee from 2010-2012 and is returning as an organizer from 2012-2014.

Course Agenda

- Introduction
- How LCSs map a problem
  - Demo of 'Classifiers'
- How LCS learn a better map
  - Demo of the main LCSs evolutionary cycle
- Organisation of a LCS
  - Demo of different concepts within LCSs
- Applications of LCSs
  - Demo of different types of LCSs
- Overview, Questions & Discussion

Introduction

- This tutorial will introduce the concept of Learning Classifier Systems (LCS)
- User-friendly in order to allow
  - Graduate students,
  - EC researchers and
  - Industry/business practitioners

who want to get up to speed with the field an easy path into using LCS to solve their complex problems.

Introduction

"LCS are a quagmire -
a glorious, wondrous and inventing quagmire, but a quagmire nonetheless" D. Goldberg '92

Not anymore!

- 30+ years research on LCS has clarified understanding, produced algorithmic descriptions, determined 'sweet spots' for parameters and delivered understandable 'out of the box' code
- This tutorial/book offers a boardwalk through the swamp

5

Introduction

- Wondrous as:

Learning Classifier Systems combine the global search of *Evolutionary Algorithms* with the local optimisation of *Reinforcement Learning* to address classification and regression problems.

- The knowledge extracted though interacting with data or embedded in an environment is human readable.
- 'Inventing' as LCS' flexible nature allows application to many domains with many types of feedback on solution progress.
- But 'swampy' as an LCS is not a one line algorithm with separable methods and easily tuned parameters.

6

Introduction

- Deliverables:
- This tutorial offers a user-friendly guide so that you will be able to proficiently implement LCS.

- It will be through explanation based on the
  - slides accompanying the book,
  - examples supported by the Python code and
  - insight/narrative from the two authors.

- Tutorial participants will have online access to the textbook, slides and code to work through the examples.

7

By The Way

- "Learning Classifier Systems" is an odd name!
- There are many Artificial Intelligence systems that learn to classifier (such as Decision Trees) that are not Learning Classifier Systems
- Learning Classifier System, abbreviated to LCS, refers to a singular/specific system, whereas Learning Classifier Systems (LCSs) refers to multiple systems or the field.
- Genetics-Based Machine Learning (GBML) is more accurate, but still not completely precise (e.g. Artificial Immune Systems are GBML, but not LCSs).
- Please accept the limitations in the name and let's explore the the concepts that underline LCSs.

8

LCSs [History 1 of 1]:

- Learning Classifier Systems are one of the earliest artificial cognitive systems.
- The early work was ambitious and broad. This has led to many paths being taken to develop the concept over the next 30 years.
- Coupling this with the fact that replicating cognition in itself is a difficult problem has led to the field being affectionately termed ‘a quagmire’ and a lack of widespread adoption.
- Early 90s simplified and essential ideas, then understanding and adaption to real-world problems.

10

Why learn using AI:

- Learning is valued by humans as it enhances our abilities to solve problems and adapt to our environment.
- Much work in research fields, such as education, psychology and neuroscience, has been conducted into how humans learn.
- With the advent of computers, humans have been interested in seeing how artificial ‘agents’ could learn. Either learning to
  - solve problems of value that humans find difficult to solve
  - for the curiosity of how learning can be achieved.

11

Assumptions for AI:

- In order for artificial learning to occur data containing the patterns to learn is needed.
- This can be through recorded past experiences or interactive with current events.
- Learning is often in the harness of a cognitive system as input data, representation, reasoning, learning and output are needed to interact with an environment

If there are no clear patterns in the data, **then** LCSs will not learn.

12

if ... then ...

The ‘if ... then ... ’ statement format essential to LCSs.

- Similarities to production rules in computer science, so the name ‘rule’ is used.
- The learned patterns are represented in the form of ‘if <this> then <that>’ rules,
- Whether
  - <conditions> <action>,
  - <state> <action>,
  - <features> <class> and many other

For convenience we will refer to conditions and actions

13

Worth of a Rule

An ‘if ... then ... ’ rule may be valid syntactically,
but we need to verify its worth

- A valid rule can quite easily encode meaningless relationships and information.
- Interestingly, the majority of valid rules are likely to contain incorrect information
- In LCSs the worth of a rule is termed ‘fitness’, due to analogies of biological fitness.

14

Fitness

Fitness is central to the operation of LCSs

It can relate to

- External effects

(e.g. the prediction of feedback from the environment)

and/or

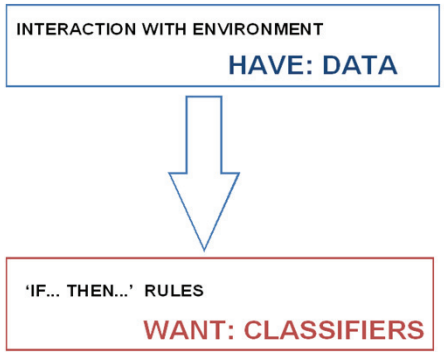
- Internal effects

(e.g. overall contribution of the rule to the system)

- Instantaneous, filtered or long-term values may be used.

15

Overview of LCSs' purpose



16

LCS as Parametric Models

- LCS are a family of methods for handling *unsupervised learning*, *supervised learning* and *sequential decision tasks* by decomposing larger problem spaces into easy-to-handle sub-problems.
- Viewpoints
  - evolutionary computation,
  - probabilistic model-based approach

- Defining question "What is an LCS supposed to learn?"

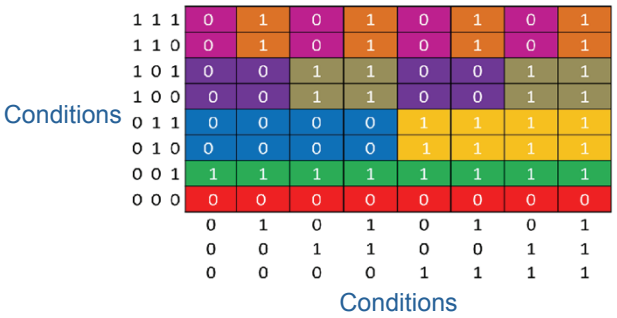
An underlying probabilistic model

Drugowitsch '08

17

LCSs as Map Generators

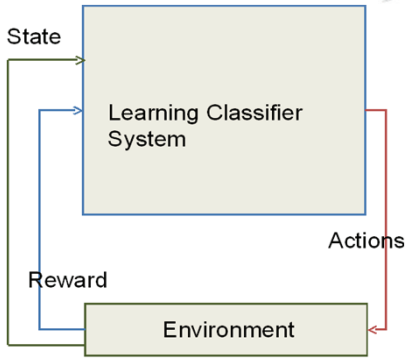
❖ The intention is to form a *map* of the problem space



Environmental interaction

Reinforcement learning: reward

Supervised learning: action



Environmental interaction

Supervised learning: The environment contains a teacher that (directly or indirectly) provides the correct response for certain environmental states as a training signal for the learning signal.

Reinforcement learning: The environment does not directly indicate what the correct response should have been. Instead, it only provides reward or punishment to indicate the utility of actions that were actually taken by the system.

Unsupervised learning: The learning system has an internally defined teacher with a prescribed goal that does not need utility feedback of any kind.

[Sutton and Barto 98]

Tasks

Search
Optimisation
Modelling
Scheduling
Knowledge-Handling
Prediction
Routing
Visualisation
Design
Querying
Learning
Game-playing
Adaptive-control
Rule-Induction
Data-mining
Diagnosis
Classification

Utilisation of LCSs

- Perpetually novel events accompanied by large amounts of noisy or irrelevant data.
- Continual, often real - time, requirements for actions.
- Implicitly or inexactly defined goals.
- Sparse payoff or reinforcement obtainable only through long action sequences.

[Booker 89]

- Main aspects found in problem domains:
  - Multimodal
  - Lack of Separation
  - High Dimensionality
  - Epistasis

Rule + Statistics = Classifier

- Classifiers are not just rules as they contain additional information:

'if < conditions > then < action >' with statistics

Prediction p Error ε Accuracy κ Fitness F

- Note: high predicting rules may not be fit if they are not consistent

- Many other additional statistics possible:

Experience, Number of offspring, Generality, Numerosity

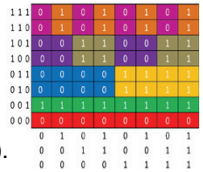
Fitness

- Fitness describes the worth of a rule.
  - embody the past success of the rule
  - indicate the quality of the knowledge held
  - Indicate the utility of the rule to the system

- There are many ways to calculate fitness

$$\text{Fitness} = \frac{\text{number of correct classifications}}{\text{experience}}$$

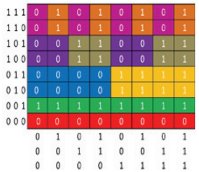
Cooperation



- One rule models a distinct part of the data (a rule covers a single niche in the domain).
- If there was only one niche in the domain, then only one rule would be needed.
- Domains of interest have multiple parts that require modelling with different rules.
- LCSs must learn a set of rules
- The rules within an LCS are termed the population, which is given the symbol [P] - the set of all rules in the population.

- The rules within a population cooperate to map the domain

Competition



- Ideally, there would only be one unique and correct rule for each niche
- Number of rules would equal number of niches

- No prior knowledge, so each rule must be learnt.
- LCSs allow multiple, slightly different rules per niche

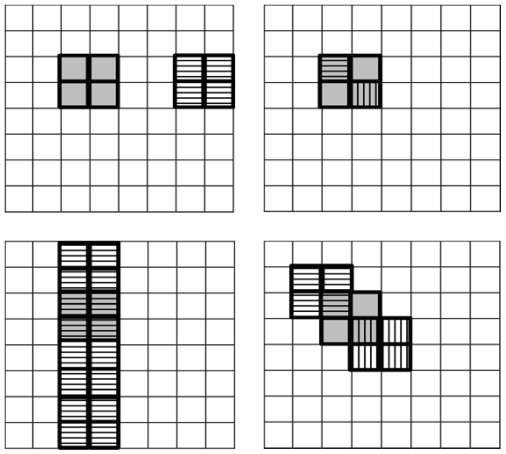
Multiple hypotheses are available to find the optimum rule

- Each rule 'covers', i.e. describes, its part of the search space.

- The rules within a niche compete to map the domain.

26

Cooperation & Competition



Grey represent ideal niche.

Which is the most useful plausible rule (stripes)?

27

Learning

- 'learning' has a very useful definition

"Learning is constructing or modifying representations of what is being experienced" Michalski et al., 86.

- LCSs need to experience the domain in order to learn.
  - embodied in an actual robot or
  - 'virtual' as in a software program receiving data
- Noise and dynamics within the data may impact on learning ability, but LCSs have shown robustness

- LCSs construct rules or modify existing rules in order to learn

28

Search space

- A major factor in determining the difficulty/likelihood of success.
- Every valid rule can be thought of a candidate solution, i.e. rules that satisfy problem constraints and encoding
- Note, we do not yet know if it is a good solution to the problem or not.
- Each candidate solution is a member of the set of possible solutions.
- The space of all candidate solutions is termed the 'search space' [alternative names, such as feasible set, feasible region and solution space exist, but are rarely used in LCS literature].
- The size of the search space is determined by both the encoding of the LCS itself and the problem itself.

29

Representation

- Environment input must be encoded
- LCSs can use multiple representation schemes.
  - Suited to binary input or
  - Suited to real-valued inputs and so forth...
- Consider representing the hours in a day in four bit binary
- ... the genotype 000111 is expressed as the phenotype 7
- The distance between similar genotypes (and their expressed phenotypes) in a search space is an important consideration when deciding upon the representation

30

Redundancy, irrelevance and compactness

- A search space has a number of dimensions.
- A single condition encodes a single dimension (feature).
- If the problem has multiple dimensions (features) then the LCS will need the corresponding number of conditions.
- The division between the conditions is implicit within a classifier. For example, the condition string: '100110'
  - 1x6-bit number, 2x3-bit numbers or 6xBoolean state
- Not all conditions are useful in the map

31

Don't Care

- A condition that we don't care about is given the symbol '#'

For example,

101:1 - the Boolean states 'on off on' has action 'on'

001:1 - the Boolean states 'off off on' has action 'on'

Can be encoded as

#01:1 - the Boolean states ' . off on' has action 'on'

- The ternary alphabet in the Classifier matches binary input
- In many instances, # acts as an OR function on {0,1}

32

LCS Recap

- A basic LCS consists of
  - data is clustered in a population of classifiers,
  - a set of classifiers can be interpreted as a model for the data
  - the most appropriate model is selected

[Drugowitsch 08]

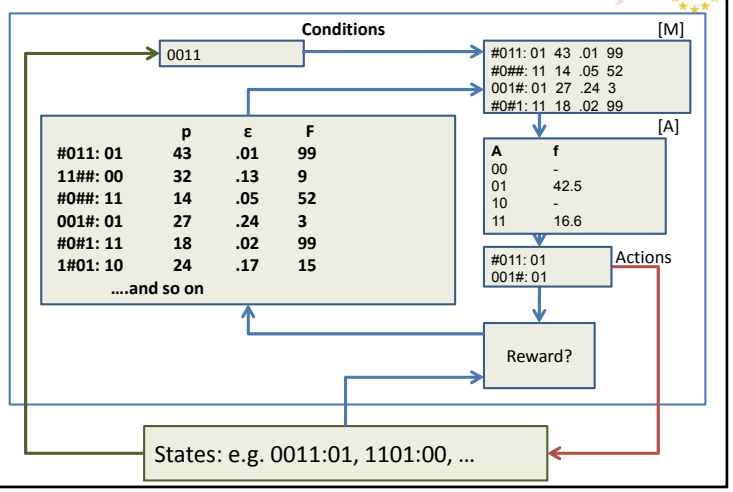
33

Steps to Evolution

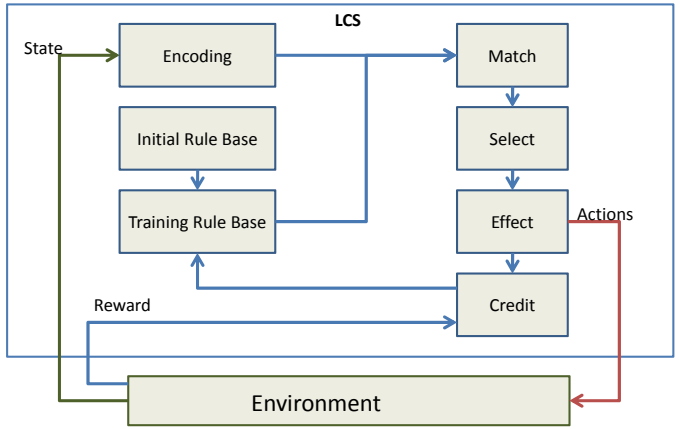
procedure evolutionary algorithm
begin
t ← 0
initialise P(t)
evaluate P(t)
while (not termination-condition) do
begin
t ← t + 1
select P(t) from P(t - 1)
alter P(t)
evaluate P(t)
end
end

37

LCSs walk through



LCSs 'Iterative' Cycle



Initialisation of an LCS

- A LCS's initial population can be:
  - Empty, Partially Full or Full
- Partially Full or Full can be:
  - Seeded (used to domain knowledge)
  - Random (avoids human bias)
- Empty
  - Utilise environmental messages (cover method)
  - Good for sparse domains and unbalanced data
  - Nowadays the method of choice, especially supervised learning.

40

Cover method

- Verb 'to Cover', or noun 'Coverage'
- Rule creation:
  - **Condition**: Generalisation of the environmental instance
  - **Action**: Known (supervised learning) or Random (reinforcement learning)

Environmental input: 101101 with corresponding action 1
with probability of generalisation (P#) of 0.33

$Rule_a$ 1##101:1 + statistics

41

Match

- Do any of the rules match the input?

Match method: for example, new input 111101 also action 1.

- **Specific**: rule conditions exactly match the input

  1----- matches $Rule_a$ 1##101:1
- **General**: 'don't care' matches all input

  -1---- matches $Rule_a$ 1##101:1

Environmental input: 111101 with corresponding action 1
Matches the previously generated rule

$Rule_a$ 1##101:1 + statistics

All matching rules placed in the match set [*M*]

42

Match

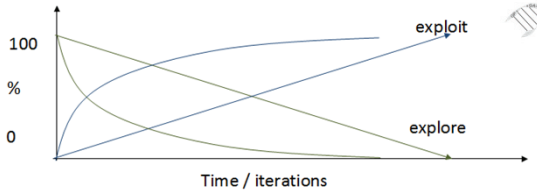
Special cases in the match method:

- No matching classifiers then invoke cover method
- Not sufficient matching classifiers then invoke cover method
- Not sufficient matching classifiers with all possible actions then invoke cover method (useful for a complete map)

- Partial match of a Classifier acceptable (e.g. 5/6 states)?
  - Low dimensional problems, this obscures class boundaries, so no.
  - May be necessary in Big Data

43

Explore vs. Exploit



- One of the biggest problems in evolutionary computation
  - When to exploit the knowledge that is being learnt?
  - When to explore to learn new knowledge?
- Annealing schemes must be set a priori without knowledge of the optimum scheme
- Often just a fixed ratio!

Select

Rules in [*M*] advocate for different actions!

- Consider the statistics of the rules

  Rule_a 1##101:1 p = 1000, ε = 0.1, F = 0.8,

  Rule_b 1#0##1:0 p = 800 , ε = 0.4, F = 0.3

- Exploit
  - Greedy (winner takes all) of p
  - Best average for each action: either p or $p_x F$
- Explore
  - Random action

Evaluate (supervised)

Experience is increased

Accuracy is calculated

$$acc = \frac{\text{number of correct classifications}}{\text{experience}}$$

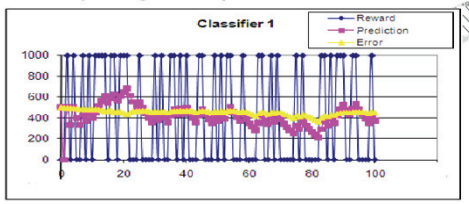
- Fitness is computed as a function of accuracy:

$$F = (acc)^{\nu}$$

Mu v used to separate similar fitness classifiers (e.g. = 10)

Evaluate (reinforcement)

- Recency weighted update



- Widrow-Hoff update: learning rate β

  value_{new} = value + β x (signal - value)
- Filters the 'noise' in the reward signal

  β = 1 the new value is signal, β = 0 then old value kept

Evaluate (reinforcement)

- Classifier considered accurate

  if error < tolerance, otherwise scaled.
- Accuracy relative to action set
- Fitness based on relative accuracy

$$p \leftarrow p + \beta(R - p),$$

$$\varepsilon \leftarrow \varepsilon + \beta(|R - p| - \varepsilon),$$

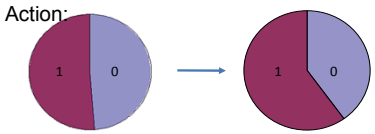
$$\kappa = \begin{cases} 1 & \text{if } \varepsilon < \varepsilon_0 \\ \alpha(\varepsilon / \varepsilon_0)^{-\nu} & \text{otherwise} \end{cases},$$

$$\kappa' = \frac{\kappa}{\sum_{x \in [A]} \kappa_x},$$

$$F \leftarrow F + \beta(\kappa' - F)$$

Select

- Rules differentiated based on fitness value



- In roulette wheel selection often raise to a power (which needs setting)
- Setting the power too high leads to local optimum at the start of training

50

Select

- Rules differentiated based on fitness rank

Action	Rank
1	2
2	3
4	4
6	1

- Rank each action in order of 'prediction'
- Chosen number of actions selected randomly
- Best action is effected
- This is rarely used to select for effect
- Often used for select for reproduction

51

Rule discovery

- **When to learn**
  - Too frequent: unsettled [P]
  - Too infrequent: inefficient training

rank based or relative rating

- **What to learn**
  - Most frequent niches or
  - Underrepresented niches

- **How much to learn**
  - How many good rules to keep (elitism)
  - Size of niche

52

Set-based Niching

- Update and creation of classifiers may occur in one of three ways:

1. Panmictic [P] 'throughout the population'
   [SCS Goldberg 89]
2. Match Set [M] restricted to the match set only
   [ZCS Wilson 94]
3. Action Set [A] restricted to the action set only
   [XCS Wilson 95]

53

Rule Discovery

Needed to create hypothesised better rules from existing rules & genetic material:

- Genetic algorithm
  - Original and most common method
  - Similar to the niched GAs method
  - Well studied (if not well understood theory)
  - Stochastic process
- Estimation of distribution algorithms
  - Sample the probability distribution, rather than mutation or crossover to create new rules
  - Exploits genetic material
- Bayesian optimisation algorithm
  - Use Bayesian networks
  - Model-based learning

Use any directed method to find new rules!

Mating – Crossover

GA: r_1 = 00010001
r_2 = 01110001

Set crossover point

GA: r_1 = 00010001
r_2 = 01110001

Applying single crossover point

GA: r_1 = 00010001
r_2 = 01110001

GA: c_1 = 01010001
c_2 = 00110001

Crossover Complete

Many variations of crossover possible:
- two point crossover
- multipoint crossover
- x-dimensional crossover

Mating – Mutation

GA: 00010001

Randomly select bit to mutate

GA: 10010101

Mutation

Mutation complete

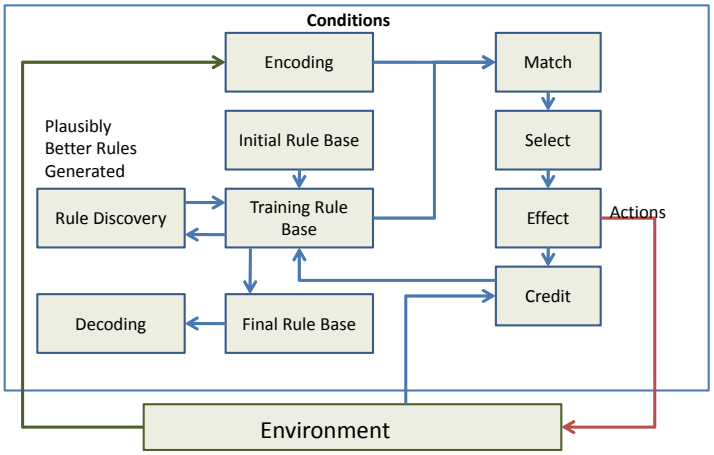
GA: 10010001

Deletion

- No deletion
  - Population grows without bound, which reduces set pressure
  - Waste memory and takes time so not often used
- Panmictic deletion
  - Most common technique based on inverse fitness roulette wheel
  - Complements set pressure
- Genotypic deletion based on generality
  - Adds bias, hard to set up and control

Advanced deletion schemes – see later

Learning Classifier Systems



Michigan Approach

- Individual is a classifier
- Cooperation and Competition between every classifier
- Population holds all individuals including incorrect, overgenerals, too specific rules

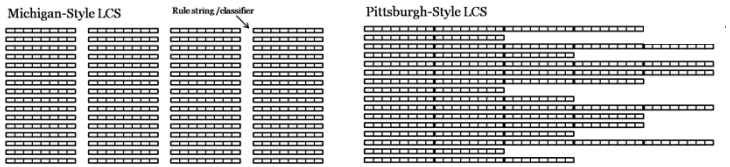
e.g. 3-bit MUX

C	A	F	…..
##1	1	50	
1#1	1	100	
11#	1	10	
000	0	89	
00#	0	100	

…

Selection based on fitness and / or prediction of individual

Major LCS-Styles



- Entire population is the solution
- Learns iteratively
- GA operates globally

- Single rule-set is the solution
- Learns batch-wise
- GA operates between rule-sets

Pittsburgh Approach

- Individual is a rule set, i.e. multiple rules
- **Competition** between every rule set to breed (Cooperation is explicit within individual)
- Population holds each unique rule set

e.g. 3-bit MUX

CA	CA	CA	F	…..
##1:1,	1#1:1,	00#:0	89	
11#:1,	000:0,	1#1:1	45	

…

- Selection based on fitness and / or prediction of set and then match of rule

Much debate on best approach!

Rights + Wrongs

- Accuracy measures the inverse of predictive error
- What happens if you are always wrong?
- If you can predict this accurately, then you are 100% accurate!
- In some cases, e.g. MUX, this doubles the rule base

C	A	P	F …..
01#	1	100	100
00#	0	100	100
1#1	1	100	100
1#0	0	100	100
01#	0	0	100
00#	1	0	100
1#1	0	0	100
1#0	1	0	100

64

Deferred reward

- Prediction *p* is updated as follows:

$$p \leftarrow p + \beta[r+\gamma \max P(s',a') - p]$$

where γ is the discount factor
r is reward, β is learning rate
s is state, a is action

- Compare this with Q-learning

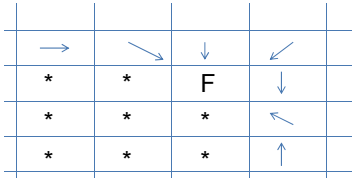
$$Q(s,a) \leftarrow Q(s,a)+\alpha[r+\gamma Q^*(s',a')-Q(s,a)]$$

where α is learning rate

65

Path Habits

- Delayed rewords cause problems
- Move NW strengthened early
- Becomes basin of attraction
- Explore does not find E move!



66

Messy Coding

- Useful for big data, i.e. many features
- Derived from Messy GAs
- Instead of using a don't care symbol can remove the feature from the condition
  11##0:1 shorten to 110:1 with encoding
- Improves transparency, reduces memory and speeds processing
- Used in bio-informatics based LCS, e.g. BioHEL and GAssist by J.Bacardit, and scalable LCS, e.g. XCSCFC

67

Choice of Encoding

Euclidean and Hamming distances alter search space

Integer	Binary	Gray	Enumerated
0	000	000	0000000
1 :1	001:1	001 :1	0000001 :1
2 :1	010:2	011 :1	0000011 :1
3 :1	011:1	010 :1	0000111 :1
4 :1	100:3	110 :1	0001111 :1
5 :1	101:1	111 :1	0011111 :1
6 :1	110:2	101 :1	0111111 :1
7 :1	111:1	100 :1	1111111 :1

Encoding :Hamming distance

How to encode the range: 0 → 3 and 0 → 4 ?

68

How to Match

Consider the message: 110001

Alphabet	Rule	Match
Ternary	1#1000 : 0	1#1000
Ternary	11#00# : 0	11#00#
Integer	111000 u : 0 101000 l	1#1000
Integer	111001 u : 0 110000 l	11#00#
Real	$1_0\,1_0\,1_0\,0_2\,0_3\,0_9$ u : 0_0 $0_5\,0_0\,0_7\,0_0\,0_0\,0_0$ l	1#1000
Real	$1_0\,1_0\,1_0\,0_2\,0_3\,1_0$ u : 0_0 $0_5\,0_0\,0_0\,0_0\,0_0\,0_9$ l	11#00#

70

Integer and Real Encodings

- Encoding suited to the environmental message
  - using upper and lower bounds:

Alphabet	Rule	Match
Integer	111001 u : 0 110000 l	11#00#
Real	$1_0\,1_0\,1_0\,0_2\,0_3\,0_9$ u : 0_0 $0_5\,0_0\,0_7\,0_0\,0_0\,0_0$ l	1#1000

[could use centre and spread, but this assumes a Gaussian distribution and recombination more difficult to implement]

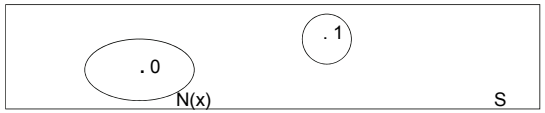
- For each allele *a*, lb ≤ x ≤ ub, to give match.
- Could use '<' instead of '≤' but LCSs determine the correct bound automatically
  e.g. 0 ≤ x ≤ 5 is equivalent to 0 ≤ x < 5.01

Mutating at the Limits

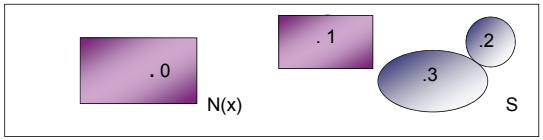
- Crossover point can either be
  - between alleles or in the middle of an allele.
- Mutation increases/decreases either or both bounds.
- Repair is occasionally needed to check that: lb < ub
- Note that most bounds have a limit, e.g. WBC: $0 \le a \le 10$
  - We decide to mutate the lower bound of a '#' $0 \le x \le 10$
  - If decrease by 10% of range to -1, repair back to 0.
  - If increase by 10% of range to 1, do not repair as valid!

Thus some alphabets have a specificity bias

Hyper Partitioning



We have a sparse search space with two classes to identify [0,1]
It's real numbered so we decide to use bounds: e.g. $0 \leq x \leq 10$, which works fine in this case...



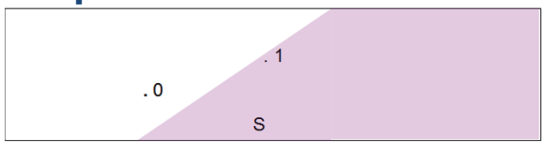
We form Hypercubes with the number of dimensions = the number of conditions
Approximates actual niches, so Classes 2 & 3 difficult to separate with this encoding, so use Hyperellipsoids

Alternative representations

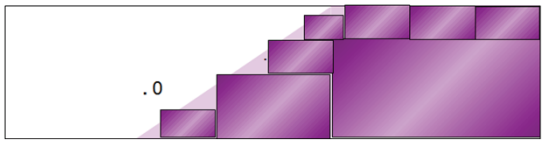
Many other representations available

- Artificial neural networks
- Fuzzy logic/sets
- Horn clauses and logic
- S-expressions, GP-like trees and code fragments.
  - Is a LCS with S-expressions not just GP? NO!
  - How to tailor functions without introducing bias?
  - How to identify building blocks of Subexpressions?
  - When are two Subexpressions equivalent?
- Is trade-off between reduced problem search space to increased alphabet search space worth it?

Oblique domains



We have a search space with only two classes to identify: 0 & 1
It's real numbered so we decide to use bounds: e.g. $0 \leq x \leq 10$



We form Hypercubes / Hyperrectangles, but these are not often suited to oblique domains
Imagine sine wave domains…..

Gracefulness

- Need to introduce new rules into population
  - Zero fitness: never selected
  - Full fitness: always selected destabilising existing rules (especially if new rule is poor)
  - Parents average: often unrepresentative
- Moyenne Adaptive Modifee followed by Widrow-Hoff update
- MAM: Simple time average from start [1/β iterations]
- WH: Recency weighted average

[Have to set when recency becomes important
and how many times steps is recent?]

75

Overgenerals

[undesired inaccurate classifiers]

- When additional reward offsets any additional penalty
- Strength-based fitness is more prone to overgenerals
- Accuracy-based fitness is less prediction orientated

Want 10011###1:1 get 10011####:1, where 10011###0:0

- Can occur in unbalanced datasets or
- where the error tolerance ε_0 is set too high.

77

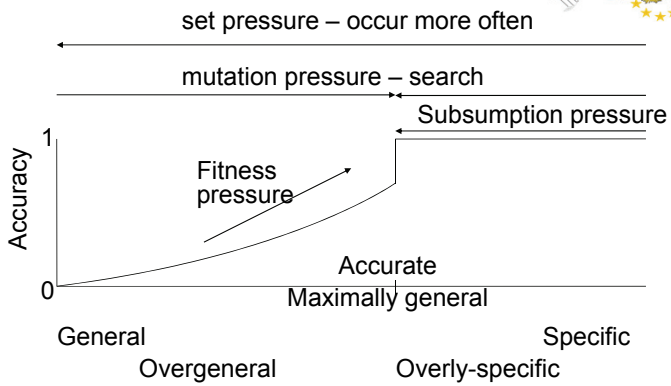
Subsumption deletion

- In sparse environments over-specific rules can take over population

  Want 10011###1:1 get 10011#011:1, 100111111:1, …
- Starvation of generals, so delete specific 'sub-copies'
- Need accurate rules first: how to set level of accuracy (often not 100%)
- GA or action set [A] subsumption?
- Effect of numerosity?

78

LCSs Pressures



Adapted from Butz '10

79

Fitness Pressure

- Fitness pressure is fundamental to evolutionary computation: "survival of the fittest"
- Fitter rules assumed to include better genetic material,
- Fitter rules are proportionately more likely to be selected for mating,
- Genetic material hypothesised to improve each generation.
- Fitness measures based on error or accuracy drive the population to rules that don't make mistakes
- Favours specific rules (cover less domain)
- Fitness measures based on reward trade mistakes for more reward
- Favours general rules (cover more domain)

80

Set Pressure

- Set pressure is related to the opportunity to breed,
- Does not occur in panmictic rule selection
- Need Niching through [M] or [A] rule discovery
- Class imbalance affects set pressure

- Set pressure is more effective when replacing 'weaker' rules
- Often panmictic deletion, thus one action can replace a different action
- To prevent an action type disappearing, relative fitness is used (rare rules have high relative fitness and so breed)

- Rules that occur in more sets have more opportunity to be selected from mating
- Favours general rules

81

Subsumption Pressure

Subsumption occurs where one rule subsumes another, i.e.:

Rule A subsumes Rule B when they both have the same action, but rule A covers rule B completely

For example:

Rule A: 11###1:1 Rule B: 1101#1:1

If rule A is completely accurate ($\varepsilon < \varepsilon_0$)
Then can delete rule B from the population without loss of performance

Favours general, but not over general, rules

Mutation pressure

- Genotypically change the specificity-generality balance

- Mutation can

Randomise:	Generalise:	Specialise:
0 ← 1 or #	0 ← #	0 ← 0
1 ← 0 or #	1 ← #	1 ← 1
# ← 0 or 1	# ← #	# ← 0 or 1

82

Subsumption Pressure

Subsumption deletion may occur in two places:

- In the action set
  - Small number of rules to check
  - Check is run often
  - Subsumed rules in population until they occur in an [A]
- In the Rule discovery
  - Large number of rules to check
  - Check is run infrequently
  - Subsumed rules never enter population
- Rules that subsume other rules
  - including copies of themselves (children can be exact copies of their parents)
  - have their numerosity increased.

84

Numerosity

Numerosity is a useful concept (trick):

- Reduces memory usage
  - Instead of population carrying multiple copies of the same classifier it just carries one copy.
  - Each rule has a numerosity value (initialised as 1)
- Protects rule from deletion
  - Stabilises rule population
- Numerosity is increased by 1
  - When subsumes another rule
  - When RD makes a copy
- Numerosity is decreased by 1
  - Rule is selected for deletion

85

Numerosity

- Numerosity affects select and update procedures:
- When calculating the fitness of an action in the select for effect procedure:
- The fitness used assumes all the classifiers in the population:

$$f = \frac{\sum F \times p \times n}{\sum F \times n}$$ Where n is the numerosity

- Macroclassifiers: all unique classifiers n ≥ 1
- Microclassifiers: all individual classifiers (n copies of macroclassifiers)
- Ratio of macroclassifiers to microclassifiers often used as a measure of training progress

86

Deletion pressure

1. Every classifier keeps an estimate of the size of the match sets in which it occurs.

The estimate is updated every time the classifier takes part in an [M]

A classifier's deletion probability is set proportional to the match set size estimate, which tends to make all match sets have about the same size, so that classifier resources are allocated more or less equally to all niches (match sets).

2. A classifier's deletion probability is as in (1), except if its fitness is less than a small fraction δ of the population mean fitness.

Then the probability from (1) is multiplied by the mean fitness divided by the classifier's fitness."

87

Deletion pressure

- The stronger the deletion bias against low fit classifiers, the more the system will tend to delete useful new classifiers before it realises how good they are.

3. A hybrid in which (1) is used until a classifier has been used on n trials, after which (2) is used.

Kovacs

n is called the delay for *t3* as it controls how long we delay the application of the low fitness penalty.

- Alternatively, classifiers may be prevented from reproduction until they have experienced a set number of trials.
  - This allows a higher initial fitness to be used, enabling simpler deletion methods to be considered.
  - However, each classifier must record number of its trials and the parent selection method becomes slightly more complex.

89

XCS

Arguably the most adopted LCS:

- Wilson 1995: Evolutionary Computation v3n2. pp1-44

- Not an acronym!
  - Although rumoured to be e**X**tended **C**lassifier **S**ystem

- Extends basic ZCS
- Niche based fitness to add set pressure
- Accuracy based LCS
- Complete and accurate mapping from inputs and actions to pay off predictions
- Maximally general subject to an accuracy criterion (aided by subsumption)

93

Complete map

- Should LCS discover:
  - The most optimum action in a niche or
  - The predicted payoff for all actions in a niche

    X x A => P (cf Q-Learning)

- The danger with optimum action only:
  - If a suboptimal rule is converged upon … difficult to discover and switch policy (CF path habits)

- The problem with predicting all actions:
  - Memory and time intensive
  - Identifies and keeps consistently incorrect action (100% accurate prediction) rules
  - Harder to interpret rule base

94

Classifier parameters

Symbol	Meaning
p	Prediction p Estimates (keeps average of) the payoff expected if the classifier matches and its action is taken by the system
ε	Prediction error Estimates the errors made in the predictions
F	Fitness Denotes the classifier's fitness
exp	Experience Counts the number of times since its creation that the classifier has belong to an action set
ts	Time stamp Denotes the time-step of the last occurrence of a GA in an action set to which this classifier belonged
as	Action set size Estimates the average size of the action sets this classifier has belonged to
n	Numerosity Reflects the number of micro-classifiers (ordinary classifiers) this classifier – which is technically called a macroclassifier - represents

96

Learning parameters

Symbol	Meaning
N	Maximum size of the population (In micro-classifiers, N is the number sum of the classifier numerosities)
β	Learning rate for p, ε, f, and as
α, ε_0, ν	Used in calculating the fitness of a classifier
γ	Discount factor used (in multi-step problems) in updating classifier predictions
θ_{GA}	GA threshold The GA is applied when the average time since the last GA in the lest is greater than the threshold.
χ	Probability of applying crossover in GA
μ	Probability of mutating an allele in the offspring
θ_{del}	Deletion threshold If the experience of a classifier is greater than θ_{del}, its fitness may be considered in its probability of deletion
δ	Mean fitness in [P] below which the fitness of a classifier may be considered in its probability of deletion
θ_{sub}	Subsumption threshold – the experience of a classifier must be greater than θ_{sub} in order to be able to subsume another classifier

98

Learning parameters

Symbol	Meaning
$P_{\#}$	Probability of using a # in one attribute in C when covering
$p_I\ \varepsilon_I\ F_I$	Initial Values in new classifiers
p_{explr}	Probability during action selection of choosing the action uniform randomly
θ_{mna}	Minimal number of actions that must be present in a match set [M] or else covering will occur
doGASumption is a Boolean parameter that specifies if offspring are to be tested for possible logical subsumption by parents.	
doActionSetSubsumption is a boolean parameter that specifies if action sets are to be tested for subsuming classifiers.	

99

Results Interpretation

No.	Condition	Action	n	F	e	p	exp
1	0010##0##############	0	24	0.81	0	1000	455
2	00001##############	0	23	0.92	0	0	7281
3	0010##0##############	1	22	0.79	0	0	7007
4	0101#####1##########	1	19	0.62	0	1000	7542
5	0111#######0########	1	18	0.66	0	0	7358
6	0110######1#########	1	17	0.69	0	1000	7324
7	0111#######1########	0	15	0.53	0	0	6013
8	1010##########1#####	1	14	0.64	0	1000	3841
9	00000##############	0	11	0.54	0	1000	74
10	00#01#1############	1	8	0.22	0	1000	1131
11	011#######00########	1	8	0.23	0	0	1810
12	11#0###########0#0#	0	8	0.27	0	1000	2483
13	0#000###0##########	0	8	0.29	0	1000	6435
14	00#1#1#1##########	0	8	0.23	0	0	7170
15	000#00############	1	7	0.21	0	0	193

Note: 2, 3, 5, 7, 11, 14, 15 incorrect, but accurate, so fit

Learning parameters

N	large enough so coverage only occurs initially
β	learning rate in the range 0.1-0.2
α	is normally 0.1
ε_0	is 1% of max reward prediction, i.e. 10 will reward is 1000
v	is typically 5
γ	Discount factor is 0.71
θ_{RD}	Rule discovery threshold is 25-50
χ	Crossover probabilities is not 0.5-1.0
μ	Mutation probability is 0.01-0.05
θ_{del}	Deletion of threshold approximately 20
δ	Deletion fraction often with 0.1
θ_{sub}	Subsumption threshold approximately 20
$P_{\#}$	Approximately 0.33
i	Initialisation parameters are very small, $P_i\ \varepsilon_i\ F_i$
P_{exp}	Exploration/Exploitation probability 0.5
Θ_{mna}	Minimal number of actions equal to the number of actions

100

XCSF

- **Real valued encoding:**

Concatenation of "interval predicates"

$$l_i \le x_i \le u_i$$

- **Crossover**: Two-point crossover point can be between predicates within an interval predicates
- **Mutation:** addition of an amount $\pm rand(m_0)$
  returns value from $(0, m_0]$ for lower bound
  Bounded random amount proved better than fixed amount
- **Repair** used as normal
- **Covering:** lower bound $l_i = x_i - rand_1(r_0)$

Returns value from $[0, r_0]$

XCSF: Piecewise-Linear Approximation

❖ **Linear approximation:**

The action is flexible for each condition range
(every time classifier matches it has the same action)

❖ Need to approximate the function $y = f(x)$
With $h(x)$ $$h(x) = w_0 + w_1 x_1$$

❖ Prediction is linear polynomial of the input components
Initially approximate a 1-D function:
- w_1 slope of a straight line
- w_0 with its intercept

Hyperplane approximation to $f(x)$

Cognitive Systems definition

Current robots are poor cognitive systems. Need to improve devices that we use every day and investigate medical benefits.

"Cognitive systems are natural or artificial information processing systems, including those responsible for perception, learning, reasoning and decision-making and for communication and action"

DTI Foresight initiative.

Cognitive LCS

Induction: Processes of Inference, Learning and Discovery (Computational Models of Cognition and Perception)

J. H. Holland, K. J. Holyoak, R. E. Nisbett and P. Thagard

Two psychologists, a computer scientist, and a philosopher have collaborated to present a framework for understanding processes of inductive reasoning and learning in organisms and machines. Theirs is the first major effort to bring the ideas of several disciplines to bear on a subject that has been a topic of investigation since the time of Socrates. The result is an integrated account that treats problem solving and induction in terms of rulebased mental models.

MIT Press (1 Jan **1986**) **ISBN:** 0262081601

Anticipatory LCS

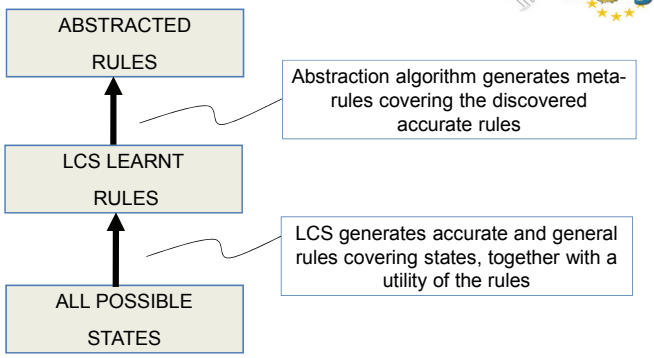
❖ Anticipations influence cognitive systems and illustrates the use of anticipations for

1. Faster reactivity
2. Adaptive behavior beyond reinforcement learning
3. Attentional mechanisms
4. Simulation of other agents
5. The implementation of a motivational module.

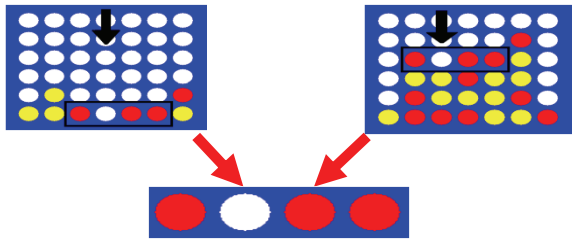
❖ A particular evolutionary model learning mechanism, a combination of
- a directed specializing mechanism and
- a genetic generalizing mechanism.

❖ Experiments show that anticipatory adaptive behavior can be simulated by exploiting the evolving anticipatory model for even faster model learning, planning applications, and adaptive behavior beyond reinforcement learning.

Abstraction



Abstraction



Connect4 is a noiseless domain so the fitness update can be simplified:

p ← p + 2 win
p ← p draw
p ← p - 2 lose

Min 0 Max 100

k ← k + 2 if prediction is correct,
k ← k – 2 otherwise

Abstracted Rules

Abstracted Rules
e.g. 'if side guide setting < width, then poor quality product

Abstraction checks for patterns in the base rules and crates and abstracted rules for each discovered pattern

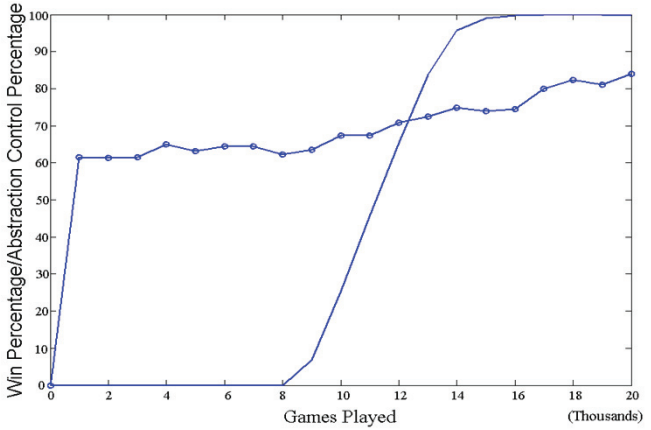
Base rules
e.g. if side-guide-setting = 80, width = 82 then poor quality product
if side-guide-setting = 79, width = 80 then poor quality product

Learning system

Raw Data
e.g. Features

'side-guide setting'	' width'	:	'product quality'
78	81	:	poor
79	80	:	poor
78	76	:	good
…			

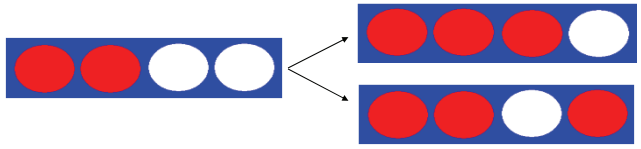
Abstraction



Staged learning

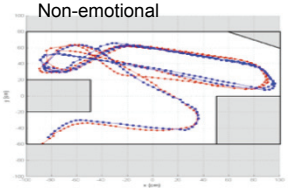
- Q-learning learns the step prior to the goal first:

- Abstraction learns the building block first

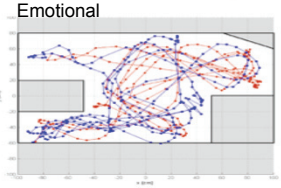


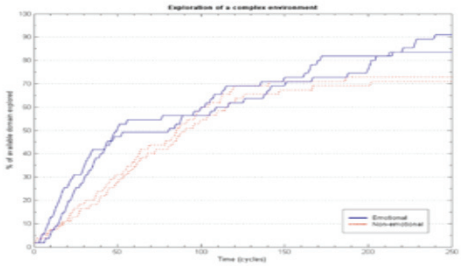
Explore efficiency

Non-emotional



Emotional



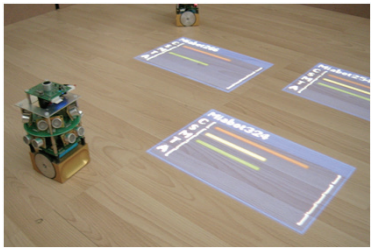


Affective learning

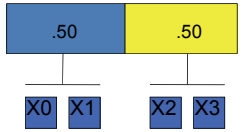
Emotion Transformer	Emotion Generator

IF SONAR > 20cm THEN
CURIOSITY ++
ELSE
CURIOSITY --
	IF CURIOSITY > 30 THEN
	SPEED += 25
	ELSE
	SPEED += 10

SENSORS → ACTUATORS



Knowledge Discovery: Manual Rule Inspection



Michigan-Style

Large rule-population

How to rank rules in order to identify best?

Imprecise rules in noisy data.

Statistical confidence for selecting attributes or rules as being important.

Pittsburgh-Style

Imprecise rules in noisy data.

Statistical confidence for selecting attributes or rules as being important.

	X0	X1	X2	X3	X4	X6	X7	X8	X10	X11	X15	X18	Class	Num.
R1	#	#	0	1	#	#	1	#	#	#	#	1	1	10
R2	1	0	1	0	#	#	#	#	#	#	#	#	1	9
R3	#	0	1	1	#	#	#	#	0	#	1	#	0	9
R4	2	1	#	#	#	#	1	#	#	#	#	#	1	6
R5	1	1	#	#	2	#	#	#	#	0	#	#	0	6
R6	0	1	0	1	#	#	#	#	#	#	#	#	1	6
R7	#	#	1	0	#	0	#	#	0	#	1	#	1	6
R8	1	0	#	#	#	#	#	2	#	#	#	#	1	5
R9	#	#	1	1	#	0	#	#	0	#	#	1	0	5
R10	#	#	1	1	#	#	#	#	0	#	2	#	0	5

Improving Interpretability and Characterization of Heterogeneity

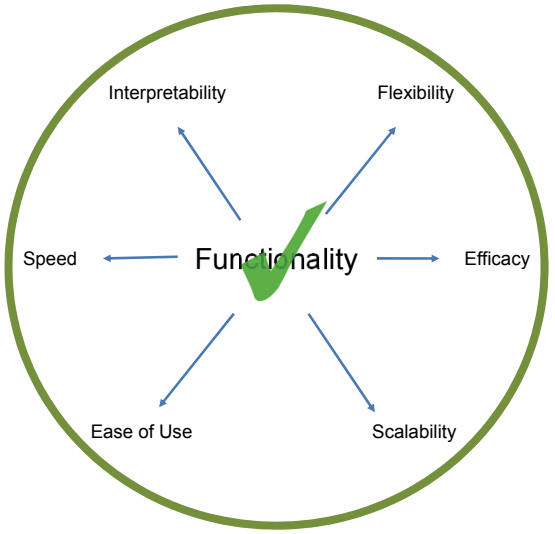
Rule Compaction Algorithm – Testing Accuracy Based

Retain a non-redundant, maximally general subset of the rule population.

Custom Fitness Calculation

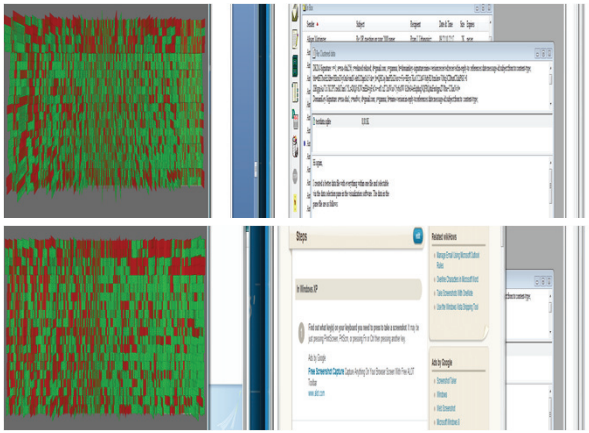
Based a rule's accuracy, generality, with and strength (age restricted)

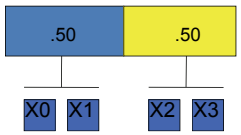
Visualization Strategies – Interpreting the Black Box



Visualizing the Rule Population:

Interpreting the Black Box – Original UCS





Attributes: 20
Predictive: 4
Non-Predictive: 16
Heritability = 0.4
MAF = 0.2
Sample Size = 1600

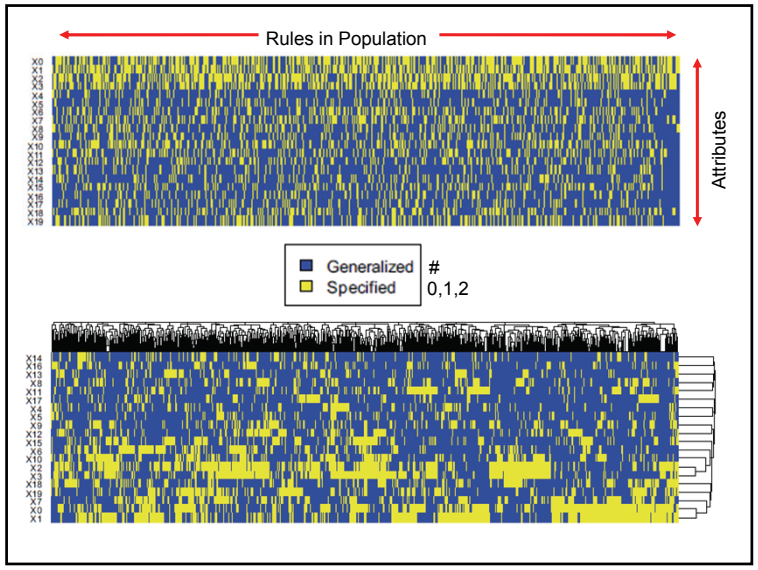
Testing Accuracy = 0.70 (p = 0.001)

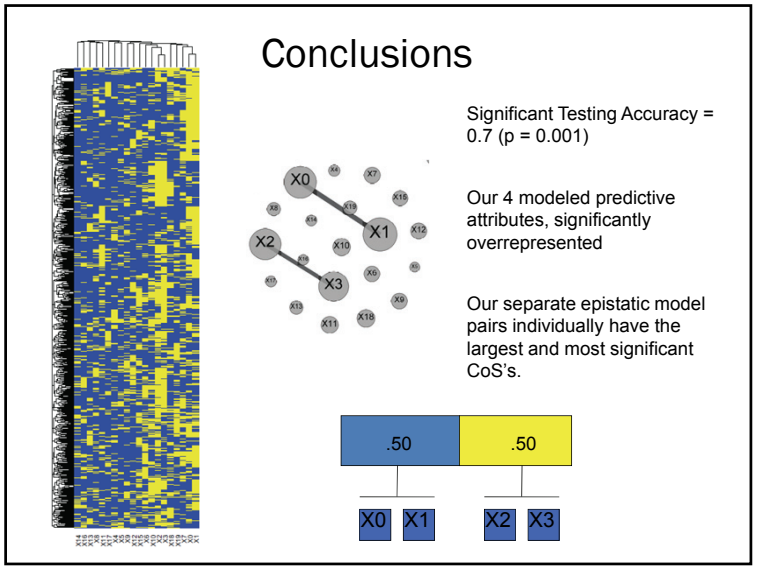
Individual Attributes

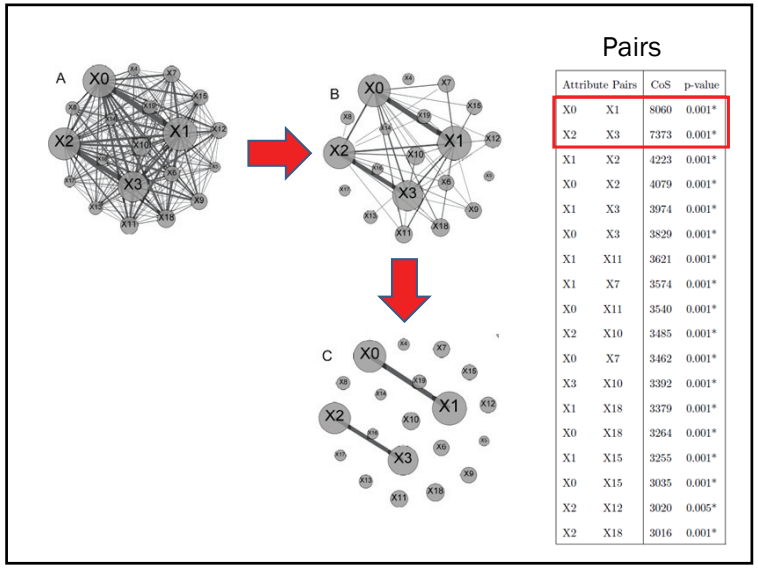
Attribute	SpS	p-value	AWSpS	p-value
X0	10885	0.001*	7589.49	0.001*
X1	11359	0.001*	7936.43	0.001*
X2	10569	0.001*	7369.84	0.001*
X3	10150	0.001*	7114.25	0.001*
X4	3863	0.999	2482.56	0.888
X5	3240	1	2090.05	1
X6	5217	0.737	3446.47	0.18
X7	5484	0.915	3647.67	0.336
X8	4429	0.95	2927.85	0.482
X9	5334	0.985	3569.25	0.484
X10	5907	0.414	3948.81	0.04*
X11	5725	0.414	3933.61	0.037*
X12	5273	1	3518.87	0.761
X13	4443	1	2854.43	0.996
X14	3709	1	2391.91	0.978
X15	5108	0.916	3425.64	0.355
X16	3613	1	2299.56	1
X17	3639	1	2302.01	0.992
X18	5933	0.629	4004.96	0.085
X19	4591	1	2940.28	0.999

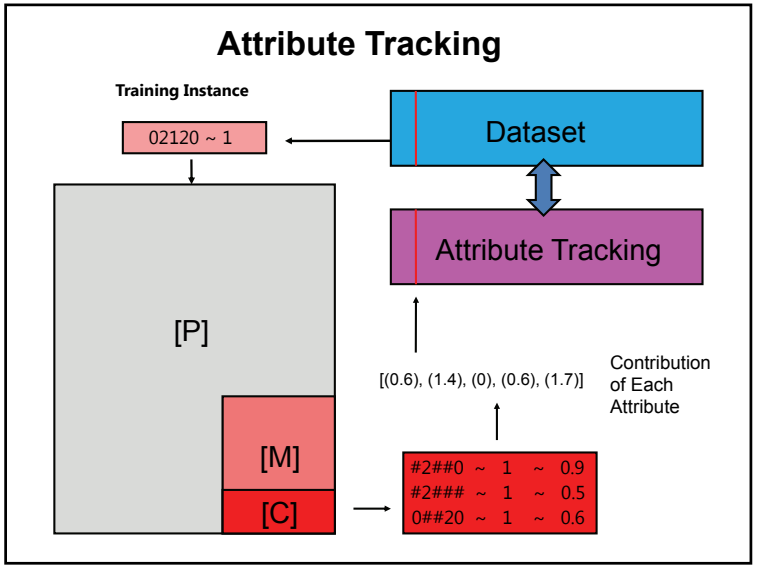
Pairs

Attribute Pairs		CoS	p-value
X0	X1	8060	0.001*
X2	X3	7373	0.001*
X1	X2	4223	0.001*
X0	X2	4079	0.001*
X1	X3	3974	0.001*
X0	X3	3829	0.001*
X1	X11	3621	0.001*
X1	X7	3574	0.001*
X0	X11	3540	0.001*
X2	X10	3485	0.001*
X0	X7	3462	0.001*
X3	X10	3392	0.001*
X1	X18	3379	0.001*
X0	X18	3264	0.001*
X1	X15	3255	0.001*
X0	X15	3035	0.001*
X2	X12	3020	0.005*
X2	X18	3016	0.001*

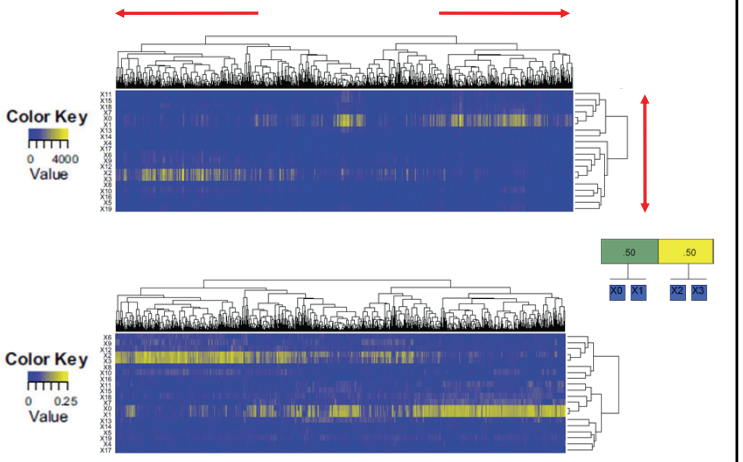

Rules in Population
Attributes
Generalized
Specified
#
0,1,2


Conclusions
Significant Testing Accuracy = 0.7 (p = 0.001)
Our 4 modeled predictive attributes, significantly overrepresented
Our separate epistatic model pairs individually have the largest and most significant CoS's.
.50
.50
X0
X1
X2
X3


A
B
C
Pairs
Attribute Pairs | CoS | p-value
X0 X1 | 8060 | 0.001*
X2 X3 | 7373 | 0.001*
X1 X2 | 4223 | 0.001*
X0 X2 | 4079 | 0.001*
X1 X3 | 3974 | 0.001*
X0 X3 | 3829 | 0.001*
X1 X11 | 3621 | 0.001*
X1 X7 | 3574 | 0.001*
X0 X11 | 3540 | 0.001*
X2 X10 | 3485 | 0.001*
X0 X7 | 3462 | 0.001*
X3 X10 | 3392 | 0.001*
X1 X18 | 3379 | 0.001*
X0 X18 | 3264 | 0.001*
X1 X15 | 3255 | 0.001*
X0 X15 | 3035 | 0.001*
X2 X12 | 3020 | 0.005*
X2 X18 | 3016 | 0.001*


Attribute Tracking
Training Instance
02120 ~ 1
Dataset
Attribute Tracking
[P]
[M]
[C]
[(0.6), (1.4), (0), (0.6), (1.7)]
Contribution of Each Attribute
#2##0 ~ 1 ~ 0.9
#2### ~ 1 ~ 0.5
0##20 ~ 1 ~ 0.6

Attribute Tracking Scores



Bladder Cancer Analysis: 10 Attributes

Testing Accuracy = 0.60 (p = 0.001)

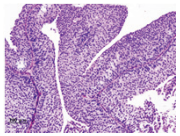
Individual Attributes

Attribute	SpS	p-value	AWSpS	p-value
XPD.751	5686	0.001*	4001.86	0.001*
XPD.312	5077	0.007*	3550.64	0.001*
pack.yr	4827	0.037*	3383.68	0.006*
XRCC1.194	4206	0.461	2818.81	0.179
age.50	4151	0.721	2800.63	0.308
male	4048	0.745	2752.61	0.358
XRCC1.399	3797	0.729	2595.78	0.427
APE1	3524	0.891	2418.03	0.667
XRCC3	3172	0.989	2166.09	0.91
XPC.PAT	2975	1.0	1994.75	0.98

Pairs

Attribute Pairs		CoS & p-value	
XPD.751	XPD.312	3367	0.001*
XPD.751	pack.yr	2757	0.001*
XPD.751	XRCC1.194	2530	0.001*
XPD.312	pack.yr	2375	0.006*
XPD.751	age.50	2345	0.016*
XRCC1.194	pack.yr	2120	0.04*
XPD.312	XRCC1.194	2105	0.046*

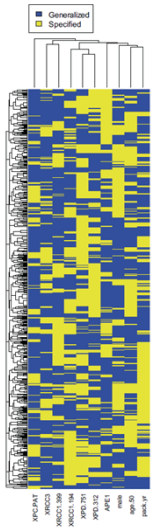
Bladder Cancer Analysis

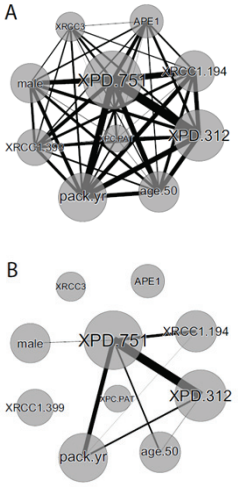


Concordance of multiple analytical approaches demonstrates a complex relationship between DNA repair gene SNPs, smoking and bladder cancer susceptibility

- Bladder Cancer
  - 4th Men (50,000), 9th Women (16,000)
- 355 Cases, 559 Controls
- 10 Attributes
  - Smoking, Age, Sex
  - 7 DNA Repair Gene SNPs
    - Xeroderma pigmentosum D (XPD)
    - Nucleotide excision repair (NER)
    - XPD 312, XPD 751
- Clinical Variables
  - Age of Diagnosis
  - Survivorship
  - Time to First Recurrence
  - Tumor Stage/Grade

Rule Population: 10 Attributes



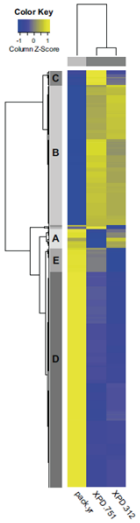


Bladder Cancer Analysis: 3 Significant Attributes

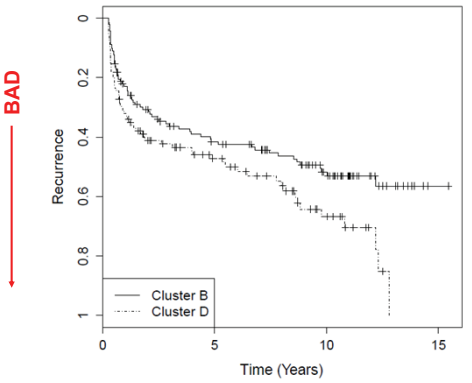
XPD 312 & XPD 751 & Pack-Years

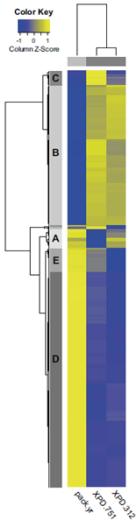
Testing Accuracy = 0.70 (p = 0.001)

- Characterizing Heterogeneity with Attribute Tracking
  - Identify Patient Subsets Characterized by Patterns of Association
  - *pvclust* – Identifies significant, stable clusters via multi-scale bootstrap re-sampling (1000)
  - 7 Significant Clusters (A-G)
  - 2 Particularly Large Clusters (B and D)
  - B – XPD SNPs yield higher tracking scores
  - D – Pack-years yield higher tracking scores

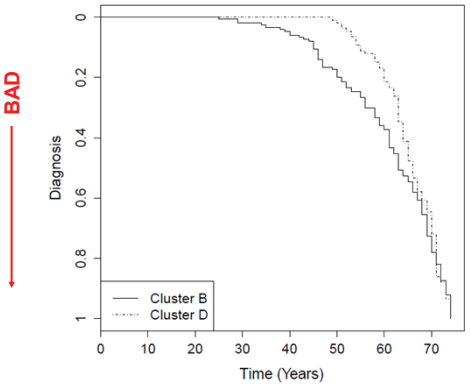


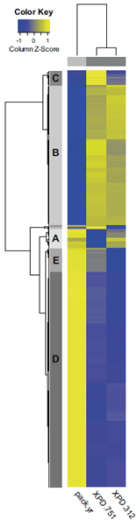
Clinical Variable Analysis: CASES: Age of Recurrence



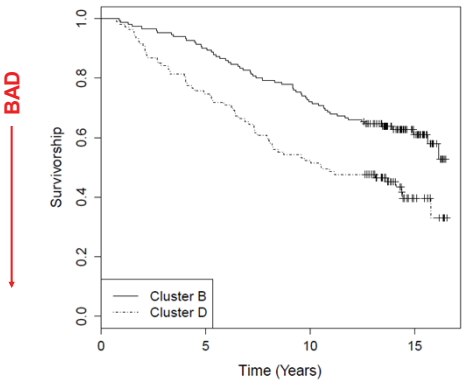


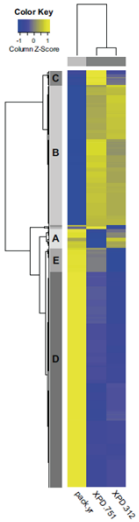
Clinical Variable Analysis: CASES: Age at Diagnosis





Clinical Variable Analysis: CASES: Survivorship





Bladder Cancer Summary

Smoking Bad

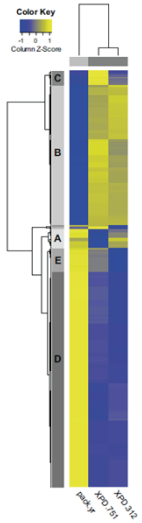
Age of Diagnosis
B – Tends to be diagnosed earlier
D – Tends be diagnosed later

Marginally Significant Difference in Recurrence
B - Later recurrence
D – Earlier recurrence

Survivorship
B – Survived longer
D – Shorter survival

Tumor Stage/Grade – Too sparse – Not Significant

Replicated Previously Implicated Risk Factors
Potential Novel Pattern of Heterogeneity



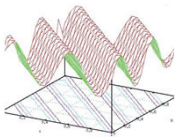
LCS Summary

- Rule-based, multifaceted, machine learning algorithms
- Global search and learning through evolution mechanism
- Local search and adaption through reinforcement learning techniques – competition with cooperation
- Multitude of flexible implementations and representations
- Practical applications as now paths through the swamp.

LCSs Applications

- **Reinforcement Learning Problems**
  Find an optimal behavioral policy represented a compact set of rules.

- **Function Approximation Problems**
  Find an accurate function approximation represented by a partially overlapping set of approximation rules.



- **Classification / Data Mining Problems**
  Find a compact set of rules that classify all problem instances with maximal accuracy.

References

- Bacardit, J., and Butz, M. V. "Data Mining in Learning Classifier Systems: Comparing XCS with Gassist", in *Advances at the frontier of Learning Classifier Systems* 276 - 290, 2007
- Booker, L. B., "Triggered Rule Discovery in Classifier Systems" *Proc. 3rd Int. Conf. on Genetic Algorithms*, page 265-274, 1989
- Butz, M., and Wilson, S. W. "An algorithmic description of XCS." in *Soft Computing: a fusion of foundations, methodologies and applications*, 162-170, 2002
- Drugowitsch, J., "Design and Analysis of Learning Classifier Systems - A Probabilistic Approach. in *Studies in Computational Intelligence*, Springer 1-239, 2008
- Goldberg, D. E., et al. "What Makes a Problem Hard for a Classifier System?" *First International Workshop on Learning Classifier Systems (IWLCS92)*, NASA Johnson Space 1992
- Kovacs, T., "Learning classifier systems resources" in *Soft Computing* 6(3-4): 240-243, 2002
- Michalski, R. S., et al., "Machine learning: A multistrategy approach" Morgan Kaufmann, 1986
- Sutton, R. S., and Barto, A. G. "Reinforcement learning: An introduction". MA: MIT Press, 1998.