The
XCS Learning Classifier System:
From Theory to Applications

Martin V. Butz
Department of Cognitive Psychology
University of Würzburg, Germany
http://www-illigal.ge.uiuc.edu/~butz
mbutz@psychologie.uni-wuerburg.de

06/26/2005

Overview

1. Learning Classifier Systems

What is an LCS?

- Characteristics:
  - Knowledge is represented by a population of classifiers (that is, a set of rules).
  - Classifiers have three main parts: condition, action, prediction.
  - Classifiers are evaluated online using reinforcement learning techniques (delta-rule).
  - New classifiers are generated online using evolutionary algorithms.

➢ An LCS is…:

A predictive, online generalizing evolutionary learning system.

Which Problems can an LCS solve?

- Reinforcement learning problems:
  - Problem modeled by *Markov decision process* (MDP)
  - Task: Learn policy to maximize reward in the long run: $\sum_{t=0}^{\infty} \gamma^t r_{t+1}$
- Classification problems:
  - Problem can be converted to an MDP
  - Direct feedback about correctness of classification
- General predictive framework:
  - Prediction of next state / next environmental properties

Knowledge Representation Example

- Rule-based representation
  - Population of classifiers represents problem knowledge.
  - Classifiers are Condition (C), Action (A), Prediction (R) rules
  - Classifier meaning: *"if condition C is satisfied then action A causes R".*
- Example:

Data Set:

σ	class
001000	1
110100	0
010101	1
100001	0
111111	1
101010	1
000110	0

Population

C	A	R
001###	0	100
1101#0	0	1000
#1#1#1	1	980
###00#	1	323
1###11	1	998
##10##	0	521
0#####	0	509
…		

Reinforcement Learning Component

- Error-based adaptation online
- Reward propagation (in reinforcement learning problems)

Database / Environment

Population

001###	0	100
1101#0	0	1000
#1#1#1	1	980
###00#	1	423
1###11	1	998
##10##	0	521
0#####	0	509
…		

σ_t **(class)**
001000 (1)

α_t
0

ρ_t
0

Match Set

001###	0	100
###00#	1	423
##10##	0	521
0#####	0	509
…		

Action Set

001###	0	100
##10##	0	521
0#####	0	509
…		

Updated A.Set

001###	0	90
##10##	0	469
0#####	0	458
…		

Genetic Discovery: Covering

- Initial classifiers are generated via *covering*
  - No current classifier available: generate a matching one.

Population

001###	0	100
1101#0	0	1000
#1#1#1	1	980
###00#	1	423
1###11	1	998
##10##	0	521
1#####	0	509
…		

σ_t **(class)**
000100 (1)

Match Set

covering

00#1#0	0	500
0#####	1	500

Genetic Discovery: Offspring Generation

- Classifiers are evolved via evolutionary algorithms
  - Steady-state genetic algorithm
  - Fitness-based *selection*
  - Genetic operators: *mutation* and *recombination*
  - *Insertion & replacement* of classifiers

Population

001###	0	105
1101#0	0	1000
#1#1#1	1	980
###00#	1	423
1###11	1	98
##10##	0	516
1#####	0	509
...		

selection

001###	0	105
##10##	0	516

mutation

001#1#	0	105
1#10##	0	516

crossover

1010##	0	310
0#1#1#	0	311

insertion & replacement

New Population

1010##	0	310
1101#0	0	1000
#1#1#1	1	980
###00#	1	423
0#1#1#	1	311
##10##	0	516
1#####	0	509
...		

How Does an LCS Work?

- Parameter estimation via gradient-based methods
  - Goal: Fast identification of current best classifiers
    - Fast and maximally accurate parameter estimates
    - Fast adaptation to population and environment dynamics
- Rule structure evolution via evolutionary methods
  - Goal: Effective search through promising solution subspaces
    - Effective selection
    - Effective local neighborhood search
    - Effective substructure propagation

Facetwise LCS Design

- Design evolutionary pressures most effectively
  - Define effective fitness measure that guides towards target solution
  - Enable effective parameter estimation
  - Ensure adequate generalization
- Ensure solution growth and sustenance
  - Proper population initialization
  - Sufficient schema supply
  - Enable schema growth
  - Ensure solution sustenance
- Enable effective solution search
  - Effective neighborhood search via mutation
  - Effective substructure search (building blocks) via recombination
  - Balance search for *local* and *global* problem structure properties

Additional Multistep Challenges

- Effective reward propagation
  - Use reinforcement learning based reward propagation.
  - Disable disruptive effects due to over-general classifiers.
- Effective behavior in environment
  - Ensure balanced problem exploration
  - Ensure effective knowledge exploitation (task dependent)
- Sampling reconsiderations
  - Unbalanced environment may cause skewed subspace occurrences

2. The XCS Classifier System

The XCS Classifier System

- Learning classifier system
- Major differences:
    - Q-learning based reinforcement learning
    - Relative accuracy-based fitness
    - Action-set restricted selection (niche selection)
    - Panmictic (population-wide) deletion

Goal:

Learn the complete *maximally accurate, maximally general* representation of the *reward map* of the problem.

Classifiers

- Condition Part C
    → *When classifier is applicable*
- Action Part A
    → *Which action to execute*
- Reward Prediction R
    → *Expected average reward*
- Prediction Error ε
    → *Estimate of mean absolute deviation of R*
- Fitness F
    → *Estimate of average action-set-relative accuracy of R*

Parameter Updates

classifier cl			reward $R(S,A,S_{+1})$	learn. rate β
condition part C	action part *A*	fitness F	discount factor γ	min. error ε_0
reward pred. R	prediction error ε		accuracy modifiers α, η	

$$cl.R \leftarrow cl.R + \beta(P(S,A) - cl.R)\left[\frac{cl.F}{\sum_{c\in[A(S,A)]} c.F}\right]$$

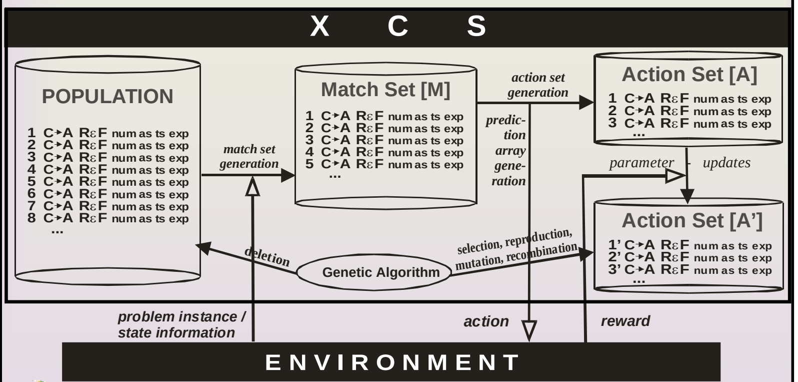
$$P(S,A) = R(S,A,S_{+1}) + \gamma \max_{A_{+1}} \frac{\sum_{cl\in[A(S_{+1},A_{+1})]} cl.F \cdot cl.R}{\sum_{cl\in[A(S_{+1},A_{+1})]} cl.F}$$

$$cl.\varepsilon \leftarrow cl.\varepsilon + \beta[|cl.R - P(S,A)| - cl.\varepsilon]$$

$$cl.\kappa = \begin{cases} \alpha(cl.\varepsilon/\varepsilon_0)^{-\eta} & \text{if } cl.\varepsilon > \varepsilon_0 \\ 1 & \text{otherwise} \end{cases} \qquad cl.\kappa' = \frac{cl.\kappa}{\sum_{c\in[A(S,A)]} c.\kappa}$$

$$cl.F \leftarrow cl.F + \beta[cl.\kappa' - cl.F]$$

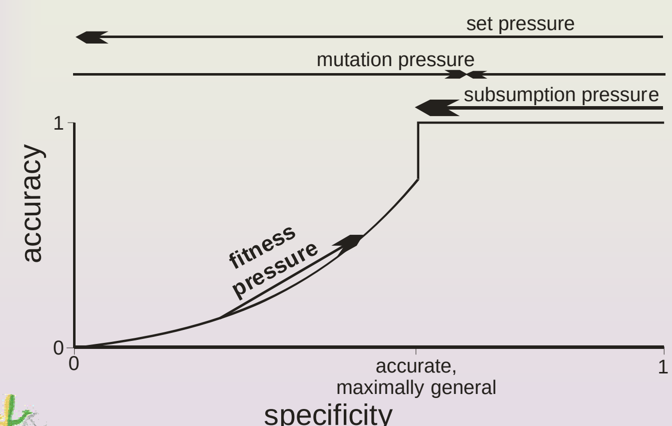
Learning Interaction



How Does It Learn? XCS Learning Pressures

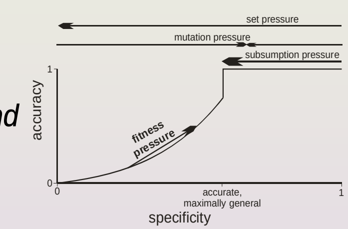
- Parameter updates identify most accurate classifiers.
- Genetic algorithm causes evolutionary pressures
  - Set pressure (reproduction of more general classifiers)
  - Fitness pressure (reproduction of more accurate classifiers)
  - Mutation pressure (diversification – specificity/generality pressure)
  - Subsumption pressure (elimination of accurate, over-specialized classifiers)

Evolutionary Pressures



Intuition for Modular Analysis

- Proper population initialization: *covering bound*
- Ensure supply: *schema bound*
- Ensure growth: *reproductive opportunity bound*
- Ensure solution support: *niche support bound*
- Enough learning time is necessary: *learning time bound*



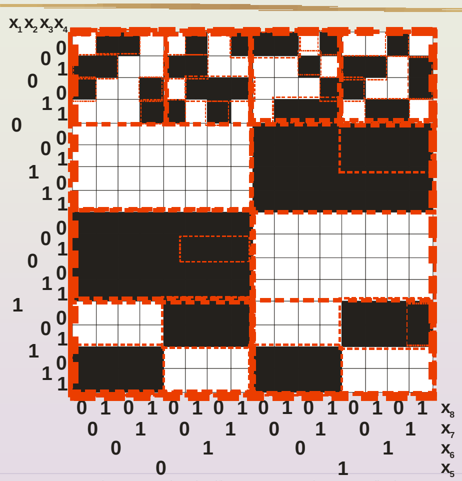
Ensuring Problem Bounds

- Problem bounds can be assured by
  - Setting initial specificity sufficiently low
  - Setting population size sufficiently high (problem difficulty)
  - Setting mutation properly (controlling specificity and search time)
  - Allowing enough learning iterations (time)
- PAC learning in k-DNF
  - Additional restrictions:
    - Uniform problem sampling (to ensure sufficient niching)
    - Balanced outcomes (to ensure accuracy guidance)
  - Linear in irrelevant attributes (given accuracy guidance) and minimal order complexity one

Learning Suitability

- XCS represents its solution by a collection of sub-solutions (that is, the population of classifiers).
- XCS learns an effective problem-dependent space partitioning in its conditions.
- Subspaces evolve to enable maximally accurate predictions.
  - Accuracy can be bounded (error threshold ε_0 and population size relation).
  - Basically any form of prediction is possible (e.g. reward, next sensory input, function value).

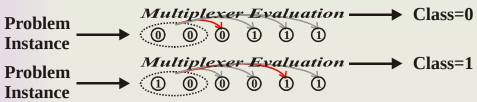
XCS Learning Structure



3. Performance Demonstration

1. Learning Classifier Systems
2. The XCS Classifier System
3. Performance Demonstration
   1. Classification Problems
   2. Function Approximation Problems
   3. (Multistep) Reinforcement Learning Problems
4. Towards Future Applications
5. Summary & Conclusions

XCS in 6-Multiplexer Problem

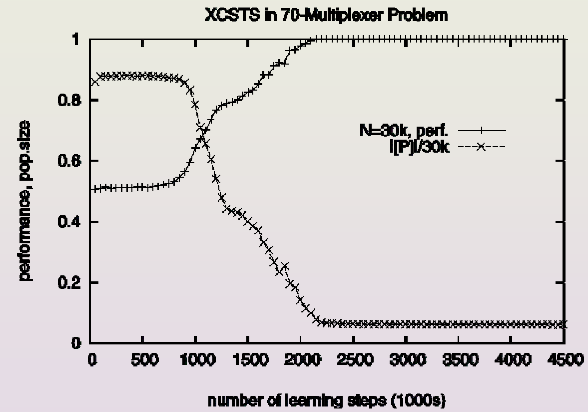


Problem instance	Class
000000	0
001000	1
000111	0
011011	0
101101	0
100010	1
100101	0
110000	0
…	…

Optimal solution representation

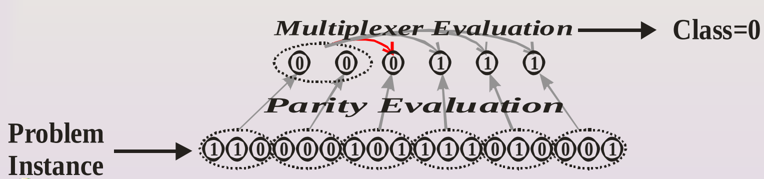
C	A	R	ε	F
000###	0	1000	0	1
000###	1	0	0	1
001###	0	0	0	1
001###	1	1000	0	1
01#0##	0	1000	0	1
01#0##	1	0	0	1
01#1##	0	0	0	1
01#1##	1	1000	0	1
10##0#	0	1000	0	1
…	…	…	…	…

XCS Performance in MP 37

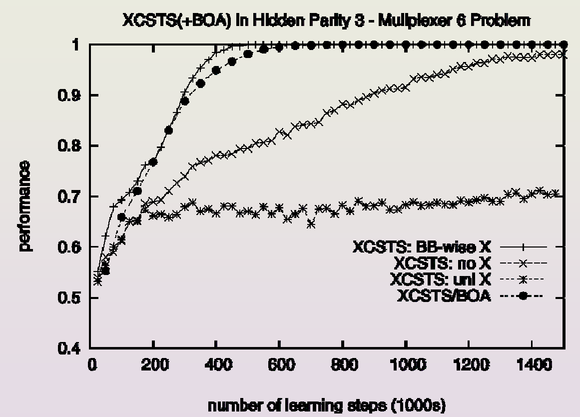


Hierarchical Classification Problem

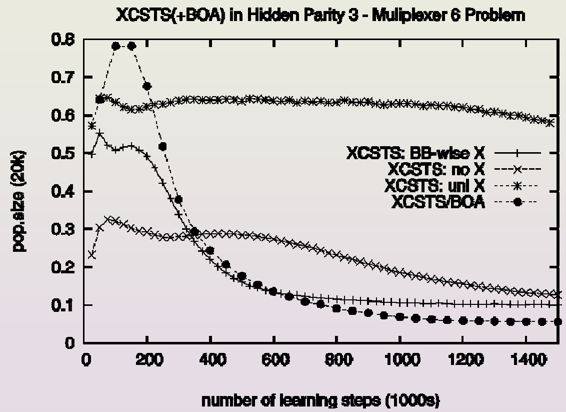
- Hierarchical problems with lower order BB structures
- BB structures are re-used on the higher level to derive problem class.
- Example: Hierarchical 3-parity, 6-multiplexer problem:



XCS/BOA Performance



XCS/BOA Performance



Real-World Datasets

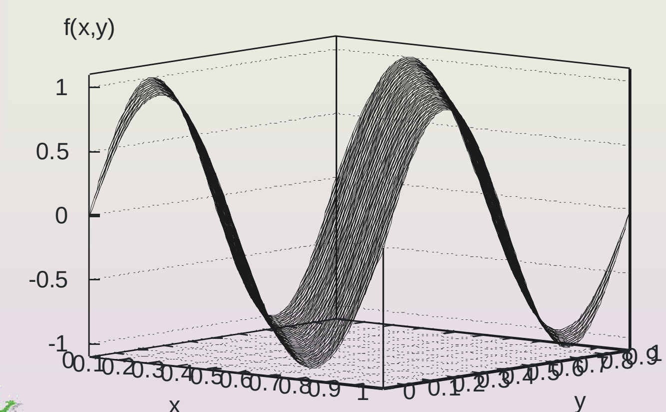
- Conditions are coded with attributes dependent on type of attribute in dataset (interval coding or Binary coding).
- Experiments in 42 datasets (from UCI and other sources)
- Comparisons with ten other ML systems (pairwise t-test)
- XCS learns competitively but it is a much more general learning system.

XCS	Maj.	1-R	C4.5	Naïve Bayes	PART	IB1	IB3	SMO (poly)	SMO (pol.3)	SMO (rad.)
99%	38/0	29/1	5/8	19/12	5/6	13/7	9/11	9/17	8/13	23/8
95%	38/0	30/1	5/9	19/12	7/6	14/7	9/15	9/18	9/14	24/9

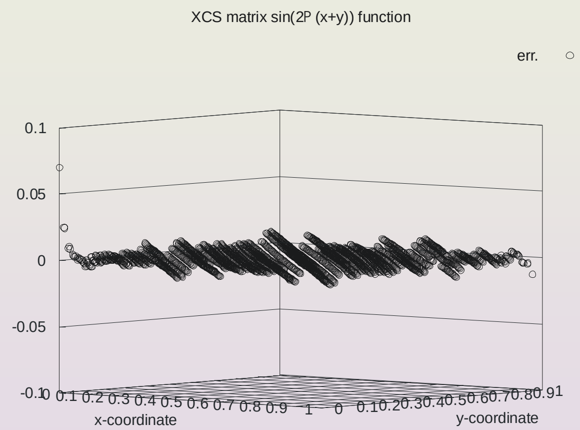
Piecewise Linear Function Approximation

- Conditions may be coded by radial bases.
  - Initialization, mutation, and crossover need to be adjusted.
  - Space partitioning into hyperspheres.
- Predictions as a linear function of the inputs
  - Usual gradient descent on weight vector
- Piece-wise linear approximations

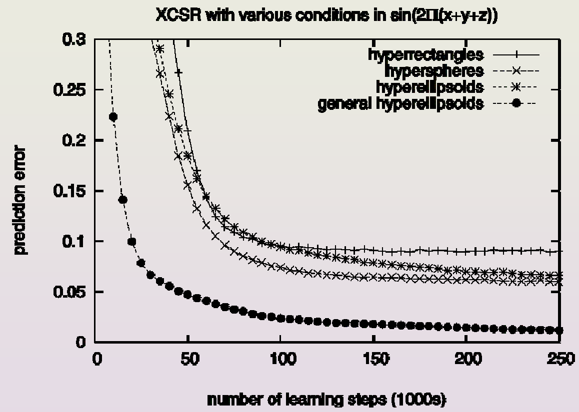
Example: Sinusoidal Function



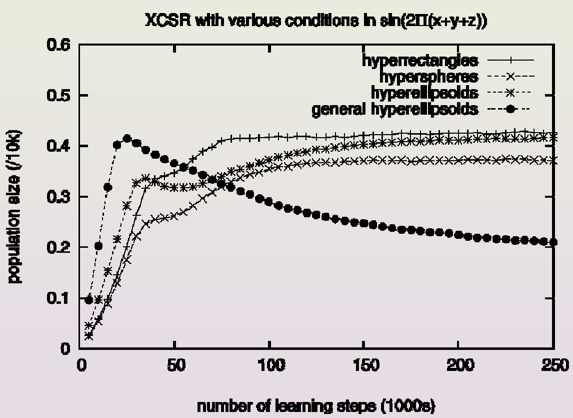
Performance in Sinusoidal Function



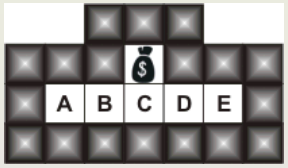
Performance in 3D Sinusoidal Function



Population Size in 3D Sinusoidal Function



XCS in Maze Problem

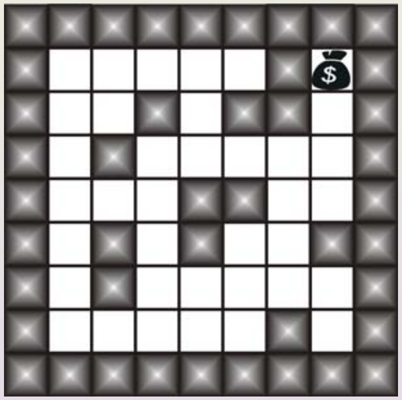


State	Sensation	↑	↗	→	↘	↓	↙	←	↖
A	11011111	.81	.81	.90	.81	.81	.81	.81	.81
B	10011101	.90	1.0	.90	.90	.90	.90	.81	.90
C	01011101	1.0	.90	.90	.90	.90	.90	.90	.90
D	11011100	.90	.90	.81	.90	.90	.90	.90	1.0
E	11111101	.81	.81	.81	.81	.81	.81	.90	.81

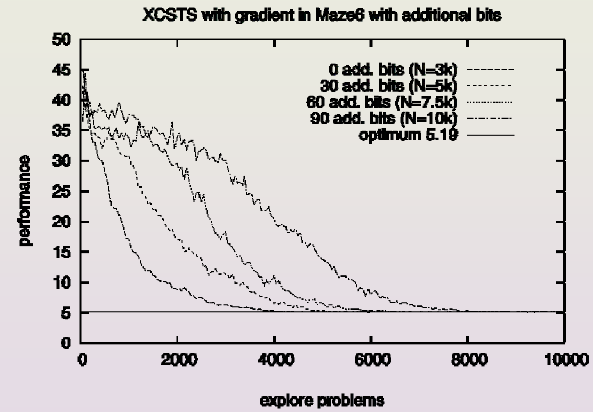
Optimal solution representation

C	A	R	ε	F
11#####1	↑	810	0	1
1#0###0#	↑	900	0	1
0#######	↑	1000	0	1
11#####1	↗	810	0	1
#10###0#	↗	900	0	1
#0######	↗	1000	0	1
##0####1	→	900	0	1
11####0#	→	810	0	1
11#####1	↘	810	0	1
…	…	…	…	…

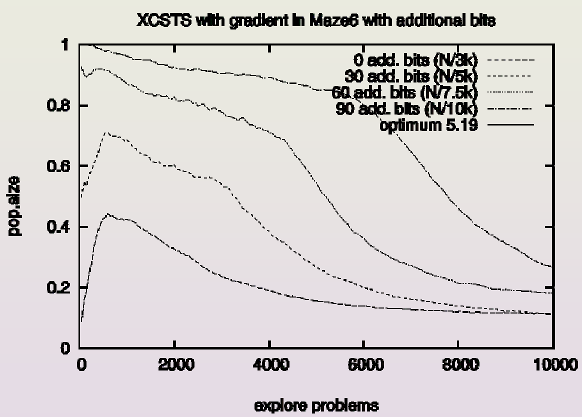
Performance Evaluation in Maze6



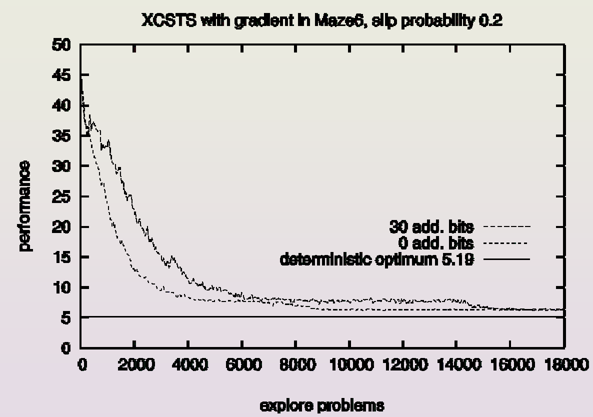
Performance Maze6 plus Irrelevant Bits



Population Size Maze6 plus Irrelevant Bits



Robustness to Action Noise



4. Towards Future Applications

XCS: A General Predictive Learner

- Space-partitioning / feature extraction for accurate predictions
- Piece-wise (constant, linear, etc.) predictions
- Predictions endowed with confidence measure
- Distinction possible between
  - space partitioning (conditions evolved by GAs)
  - generation of prediction (learned by gradient methods)

Design of XCS Structure

- Know the problem space
  - Which space partitioning should work best?
    - Radial bases
    - Hyper-rectangles
    - General ellipsoids
    - Gabor-filters
- Know the transition space
  - Which prediction type should work best?
    - Constant
    - Linear, polynomial, … prediction

Choice of Parameters and Operators

- Estimate problem complexity for population size
  - How crisp should the conditions be?
    - Probability of subspace occurrence
    - Expected complexity of accurate solution
  - How difficult will be the initialization?
- Design genetic operators accordingly
  - Which are promising neighborhoods of condition structures?
    - Design mutation operator accordingly
    - Consider effect on specificity and scale accordingly
  - How are conditions recombined most effectively?
    - Crossover constraints
    - Estimation of distribution algorithms

Multistep Considerations

- Choose effective exploration strategy
  - Random behavior during exploration
  - Additional exploratory behavior (choose action with highest information gain)
  - Learning only during exploration
- Ensure effective reward propagation
  - Use residual gradient information
- POMDP
  - Currently a big challenge
  - Detection of aliasing states might be possible

5. Summary and Conclusions

Summary

- XCS Classifier System is…
  an online generalizing, structure extracting gradient-based and evolutionary-based learning system.
- XCS represents its problem solution…
  by possibly partially overlapping sub-solutions.
- XCS can solve…
  classification, function approximation, generalizing reinforcement learning, and general prediction problems.
- Thus, XCS does:
  - Clustering for accurate predictions
  - Online learning and generalization

Conclusions

- XCS is an effective online space partitioning system.
- It is expected to be applied most effectively to problems with some of the following properties:
  - Online learning necessary, potentially with sparse reward.
  - Problem space sampling is approximately uniform.
  - Mainly non-overlapping partitions are adequate for accurate predictions.
  - Many features are expected to be irrelevant.
  - Conditionally linear (potentially action-dependent) predictions are expected.

Thank You for Your Attention

Representation of Condition

- Original XCS for binary input
  - Current complexity analysis for binary input
  - Complexity analysis extendable to other representations
- Representation “independence”
  - XCS for symbolic inputs (Messy XCS, Lanzi, 1999)
  - XCS for real valued input (XCSR, Wilson,1999)
  - XCS with integer inputs (XCSI, Wilson, 2000)
  - XCS with S-expressions (XCSL, Lanzi, 1999-2001)
  - XCS as a general function approximation tool (Wilson, 2001)

Covering Bound

- Covering-deletion loop needs to be prevented.
- Matching probability needs to be large enough.
- Specificity needs to be sufficiently small.
- Setting specificity to 1/l, population size needs to be sufficiently large.

$$\sigma[P] < 2(1-N^{-1/l}) \qquad N > -\log(1-P(\text{cov.}))\exp^{n/2}$$

Schema Bound

Schema notion:

Example:

Schema: 10**0* (order k=3, defining length d=4)

Classifier *representatives*:

10##0#,10##00, 10#000, 100000,101000,…,100#0#,101#0#

- Schema representatives of minimal order need to be present (problem dependent).
- Bounds specificity (needs to be sufficiently large).
- Population size needs to be sufficiently large.

$$\sigma[P] \ge 2n^{1/k_m}(1-(1-P(\text{representative}))^{1/N})^{1/k_m} \approx 2\left(\frac{n}{N}\right)^{1/k_m}$$

$$N \ge -n\left(\frac{2}{\sigma[P]}\right)^{k_m}\log(1-P(\text{representative}))$$

Reproductive Opportunity Bound

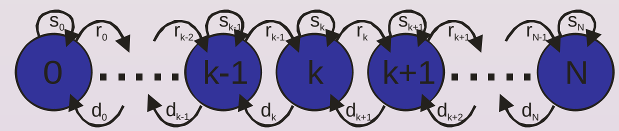
- Representatives need to be reproduced before being deleted with high probability.
- Minimal order k_m schemata need to be processed in representatives.
- Given the required specificity behaves in 1/l, the population size bound yields:

$$N(\log_2 N)^{k_m} > nl^{k_m} \qquad N : O(l^{k_m})$$

Niche Support Bound

- Learned problem representation needs to be maintained.
- Each niche underlies a Markov process of reproduction and deletion.
  - Given a niche, the number of representatives is between zero and N (maximal population size).
  - The state of a niche is its number of representatives.
- Steady-state of Markov chain can be derived.
- Probability of zero state corresponds to probability of niche loss.

$$u_k = \binom{N}{k} p^k (1-p)^{N-k} \qquad u_0 = (1-p)^N$$



Time Bound

- Given the other bounds are satisfied, we know that a solution will evolve and will be sustained.
- How long does it take to find the maximally accurate, maximally general classifiers?
  - Starting from the over-general side;
  - Assuming domino convergence (one attribute after another);
  - Need to consider:
    - Time until reproduction
    - Time until production of next best classifier (analysis considers mutation only)
  - Then, estimate time until maximally accurate solution (of order k_d) is evolved with high probability:

$$O(l2^{k_d+n})$$