Be Real! XCS with Continuous-Valued Inputs

Hai H. Dam
Artificial Life and Adaptive Robotics Laboratory,
School of ITEE,
UNSW@ADFA
Canberra, ACT 2600, Australia
z3140959@itee.adfa.edu.au

Hussein A. Abbass
Artificial Life and Adaptive Robotics Laboratory,
School of ITEE,
UNSW@ADFA
Canberra, ACT 2600, Australia
abbass@itee.adfa.edu.au

Chris Lokan
Artificial Life and Adaptive Robotics Laboratory,
School of ITEE,
UNSW@ADFA
Canberra, ACT 2600, Australia
cjl@itee.adfa.edu.au

ABSTRACT

XCS is widely accepted as one of the most reliable Michigan-style learning classifier system (LCS) for data mining. In order to handle real-valued inputs effectively, the traditional ternary representation has been replaced by the interval-based representation and the modified XCS has shown to work well. Existing interval-based representations still suffer from a few drawbacks which this paper address. In this paper, we propose an alternative approach called the Min-Percentage representation which produces comparable results to other methods in the literature with the extra advantage of overcoming the drawbacks in these methods.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

General Terms

Design

Keywords

XCS, Learning Classifier System, ternary representation, interval representation

1. INTRODUCTION

Many implementations of XCS are based on the ternary representation, which is unnatural for continuous-valued inputs when using real-world data. To overcome this problem, Wilson [5] modified XCS and introduced the Center-Spread representation. His experiments showed that XCS works well on the 6-Real-Multiplexer problem. In the Center-Spread Representation (CSR), an interval predicate takes the form (c_i, s_i) where $c_i, s_i \in [p_{min}, q_{max})$, p_{min}, q_{max} are the lower and upper bound of an interval. c_i is the center of the interval and s_i is the width of the interval from the center. The use of this representation requires a truncation when mapping a genotype to phenotype and after applying genetic operators. Therefore, the bias generated through this truncation process has been shown to be undesirable [4].

Wilson later introduced another representation called the Min-Max Representation (MMR) [6] for integer-valued inputs and showed that XCS is able to learn oblique data with integer-valued inputs. In the MMR, the interval predicate is represented as (l_i, u_i) where l_i, u_i are the minimum and maximum bounds of the interval. The MMR overcomes the bias in the CSR caused by truncation operator. However, the problem of this representation occurs after the use of Genetic Algorithm operators where it can generate an infeasible interval with $l_i > u_i$.

Stone and Bull [4] argued that the MMR can also be applied for real-valued inputs and they extended it to what they called the Unordered-Bound Representation (UBR). Stone and Bull proposed this representation to fix the problem of the MMR by allowing the interval to be represented as (p_i, q_i) where either p_i or q_i can be the maximum or minimum bound. UBR has solved the problem of generating infeasible intervals but raises another challenge.

Holland's schema theorem and the building block hypothesis [3] explain why genetic algorithm is an adaptive and efficient search method [2]. XCS depends heavily on the evolutionary learning process to drive the classifier of low accuracy to those of high accuracy [1]. However, the UBR changes the semantics of the chromosome by alternating between the min and max genes; that is, in one generation the genes representing the lower bound of an interval can become the genes representing the upper bound of an interval in the following generation. This discrepancy will generate a challenge to building blocks.

In this paper, we present the Min-Percentage representation, which performs equivalent to other representations but overcomes the previous drawback.

2. THE MIN-PERCENTAGE REPRESENTATION

To overcome the problem of the UBR, we propose the Min-Percentage Representation (MPR) which maintains the semantics of the genotype. Similar to other approaches, each attribute in the *Condition* of the classifier is represented as an interval predicate in the form of (m_i, p_i) where m_i is the minimum bound of an interval in phenotype and p_i is the proportion of the distance between the minimum and maximum bound of an interval and the distance between the upper bound and minimum bound of an interval of the phenotype. The transformation from genotype (m_i, p_i) to phenotype (l_i, u_i) is taken as follows:

$$l_i = m_i \tag{1}$$
$$s_i = p_i * (p_{max} - l_i) \tag{2}$$
$$u_i = m_i + s_i \tag{3}$$

where s_i is the distance between the lower and upper bound.

In order for XCS to work with the continuous representation, we also need to change the genetic operators such as mutation and crossover; covering function; and subsumption function. We used the same mutation, crossover, and subsumption operators used by [4]. However, because of the change in the representation, we needed to modify the covering operator.

The covering technique introduces a new classifier into the system when the population does not contain any classifier to match a current input. Similar to the UBR, the covering operator creates a classifier containing intervals (l_i, u_i) in phenotype given by:

$$l_i = x_i - R[0, s_0) \tag{4}$$
$$u_i = x_i + R[0, s_0) \tag{5}$$

where s_0 is a constant number, $R[0, s_0)$ is a random number between 0 and s_0, and x_i is an input value. After the truncation, l_i and u_i fall within the range $[p_{min}, p_{max})$, m_i is set to l_i; p_i is calculated as:

$$p_i = \frac{u_i - l_i}{p_{max} - l_i} \tag{6}$$

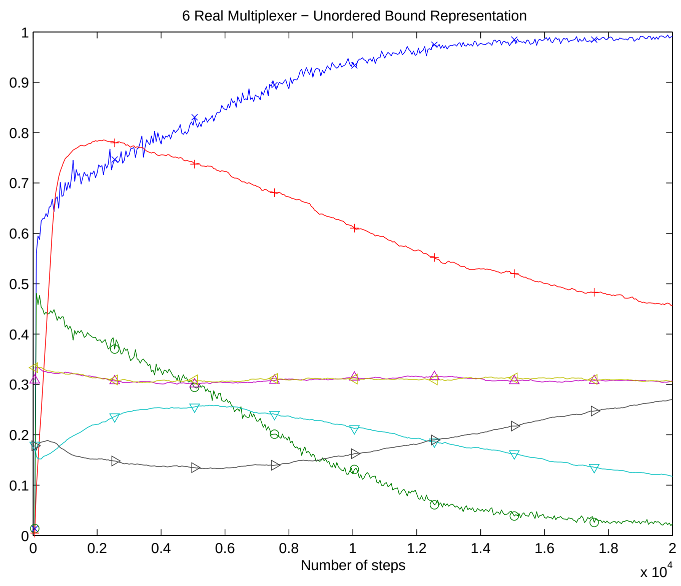
3. EXPERIMENTS SETUP

In order to verify XCS in the continuous environment, Wilson [5] modified the Multiplexer problem to Real-Multiplexer problem. Stone and Bull lately claimed that the Real-Multiplexer problem is biased towards intervals which contains a predicate. They introduced an alternative problem called the Checkerboard. We decided to test on both the 6-Real-Multiplexer and Checkerboard problems.

Stone and Bull [4] showed the better performance of the UBR when compared to other representations. Therefore, we decided to compare the MPR against the UBR.

We used the same parameter settings used by Wilson [5] and Stone and Bull [4] as follows: $N = 800, \beta = 0.2, \alpha = 0.1, \epsilon_0 = 10, v = 5, \theta_{GA} = 12, \chi = 0.8, \mu = 0.04, \theta_{del} = 20, \delta = 0.1, \theta sub = 20, p_I = 10, \varepsilon_I = 0, f_I = 0.01, \theta nma = 2, m = 0.1, s_0 = 1.0$.

All experiments presented are averaged over 30 independent runs. The random number generators for all representations are synchronized so that they all deal with the same data.



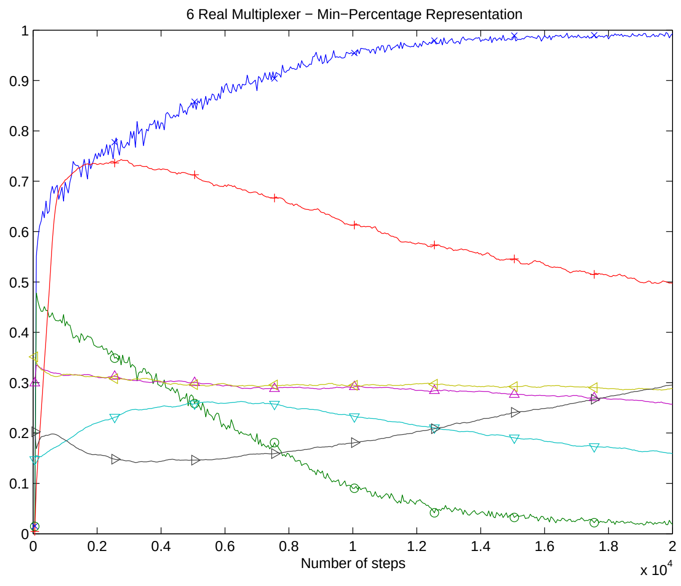
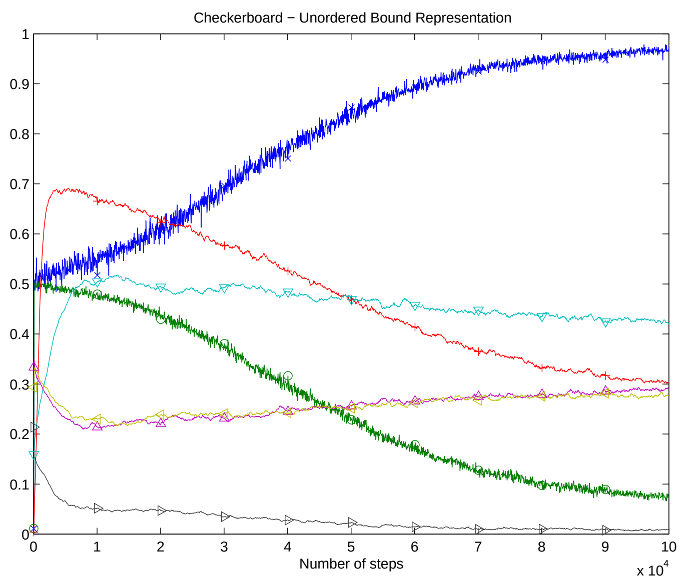


Figure 1: System Performance, System Error (o), Macro-classifier Fraction (+), Intervals within *Upper and Lower Bound* Proportion (triangle down), Intervals contain only *Lower Bound* Proportion (triangle up) , Intervals contain only *Upper Bound* Proportion (triangle left), Intervals contain both *Lower and Upper Bound* Proportion (or *Don't Care* interval)(triangle right)

4. RESULTS

Figure 1 shows the system performance of XCS on the 6-Real-Multiplexer problem using the UBR and MPR respectively. The result shows that the MPR converges a little bit faster than the UBR in the beginning, then UBR catches up and both representations achieve equivalent performance at the end. It is also shown that the MPR produces slightly more macro-classifiers, more *don't care* and *within* intervals than the UBR.



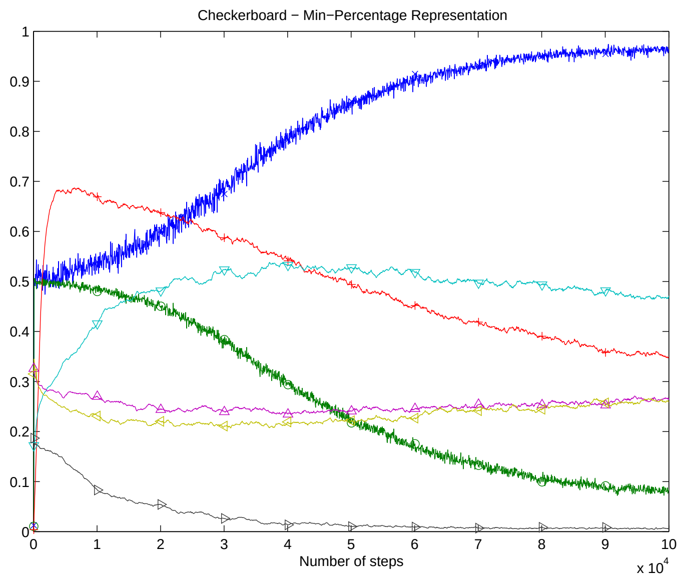


Figure 2: System Performance, System Error (o), Macro-classifier Fraction (+), Intervals within *Upper and Lower Bound* Proportion (triangle down), Intervals contain only *Lower Bound* Proportion (triangle up) , Intervals contain only *Upper Bound* Proportion (triangle left), Intervals contain both *Lower and Upper Bound* Proportion (or *Don't Care* interval) (triangle right)

Figure 2 shows the performance of XCS on the Checkerboard problem with the UBR and MPR respectively. Similar to the 6-Real-Multiplexer problem, the MPR converges quicker than the UBR up to 50,000 steps, then both obtain almost equivalent performance at the end. Again, the number of macro-classifiers of the MPR seems to be slightly higher than the UBR. Also, more *don't care* and *within* intervals are produced with the MPR than the UBR.

In general, both representations behave almost similar in both problems and achieve equivalent performance. However, the MPR maintains the semantics of the genotypes and thus is unlikely to cause discrepancies in building blocks.

5. CONCLUSION

In this paper, we introduced the Min-Percentage representation for continuous valued inputs. The tests on the 6-Real-Multiplexer and Checkerboard problems reveal that the Min-Percentage representation can perform as well as the Unordered-Bound representation but converges a little bit quicker at the beginning. Also, the Min-Percentage representation maintains the semantics of the genes all over the evolutionary run; therefore unlikely to cause discrepancy to building blocks

6. ACKNOWLEDGMENTS

Work reported in this paper was funded by the Australian Research Council Linkage grant number LP0453657.

7. REFERENCES

[1] M. V. Butz, D. E. Goldberg, and K. Tharakunnel. Analysis and improvement of fitness exploitation in XCS: Bounding models, tournament selection, and bilateral accuracy. *Evolutionary Computation*, 11(3):239–277, 2003.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addision-Wesley Publishing Company, INC., 1989.

[3] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975. Republished by the MIT press, 1992.

[4] C. Stone and L. Bull. For real! XCS with continuous-valued inputs. *Evolutionary Computation*, 11(3):299–336, 2003.

[5] S. W. Wilson. Get real! XCS with continuous-valued inputs. In P. Lanzi, W. Stolzmann, and S. Wilson, editors, *Learning Classifier Systems, From Foundations to Applications, LNAI-1813*, pages 209–219, Berlin, 2000.

[6] S. W. Wilson. Mining oblique data with XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Proceedings of the Third International Workshop (IWLCS-2000), Lecture Notes in Artificial Intelligence*, pages 158–174, 2001.