Towards Final Rule Set Reduction in XCS: A Fuzzy Representation Approach

Farzaneh Shoeleh
Computer Science and Engineering Dept.
Shiraz University, Shiraz, Iran
shoeleh@cse.shirazu.ac.ir

Ali Hamzeh
Computer Science and Engineering Dept.
Shiraz University, Shiraz, Iran
ali@cse.shirazu.ac.ir

Sattar Hashemi
Computer Science and Engineering Dept.
Shiraz University, Shiraz, Iran
s_hashemi@shirazu.ac.ir

ABSTRACT
Generalization is the most challenging issue in XCS research area. One of the main components of XCS managing to remedy this issue is knowledge representation. In this paper, a knowledge representation based on fuzzy membership function offering certain and vague regions is described. We extend the Michigan learning classifier system using this approach to be improved in terms of both performance and interpretability. The contribution of this paper is three-folds: 1) updating main parameters of classifiers based on their certainty factor in matching of incoming data, 2) enhancing essential components of XCS to be compatible with such fuzzy representation schema and 3) proposing a novel rule set reduction method named Reduction based on Least Reward Prediction (RLRP) to improve the interpretability of the evolved model. Furthermore, an inference methodology which is compatible with RLRP is suggested to maintain the similar performance. The obtained results are promising due to the effectiveness of proposed method in dealing with real world problems. Furthermore, the proposed reduction method can upgrade the interpretability of final rule set by boiling its size down by 94% on average while slightly degrading the prediction accuracy.

Categories and Subject Descriptors
I.2.6 [**Learning**]: Concept learning, Knowledge acquisition

General Terms
Algorithms

Keywords
Knowledge Representation, Learning Classifier System, Reduction Method, XCS.

1. INTRODUCTION
Learning classifier systems (LCSs) are evolutionary learning mechanisms that combine the general principles of Darwinian evolution with the power of the cognitive learning paradigm [1]. LCSs are on line rule based systems which their main aim is using internal evolutionary algorithm to evolve a population of classifiers which are usually represented as rules named classifiers as a solution to a given problem. The accuracy based classifier system, namely the eXtended Classifier System (XCS) [2], is currently considered as millstone of learning classifier systems due to its effectiveness in data analysis and its success in applying to varieties of learning problems. It can be concluded that the effectiveness of XCS as a machine learning paradigm is that "*XCS was the first classifier system to be both general enough to allow applications to several domains and simple enough to allow duplication of the presented results*" [3].

A rule based system such as XCS is able to solve a problem efficiently if its rule set can cover the whole problem space properly and also each rule makes an effective decision. Therefore, it can be concluded that rule representation, i.e. knowledge representation, has an essential role in rule based systems. In addition, one of the most important goals in the extensions of XCS is to achieve a compact and accurate solution for a given problem [4]. It is obvious that achieving these goals mainly depends on the representation method which is used to cover the problem space.

One of the main challenging issues in proposing a knowledge representation technique is how classifiers of population should cover the problem space to define the problem regularities accurately. In addition, how a classifier can match an incoming instance is an important factor in choosing the best action. The last few decades have seen the increasing interest in using fuzzy logic and fuzzy theory as a powerful mechanism to create maximally general and accurate rule set. The benefit of using fuzzy theory is not only handling uncertainty and imprecision but also avoiding to evolve large number of rules that may hamper the readability of the rule set. In one of the recent researches [5], a Michigan style learning classifier system named XCS-HT is introduced. The main aim of that study was proposing a new knowledge representation based on presenting two kinds of regions, namely certain region and vague region, in the condition part of each classifier to match the incoming instance with different levels of certainty. It was shown that such representation helps XCS-HT to evolve a set of classifiers which can cover problem space in an effective manner with minimum redundancy [5]. In other words, the evolved rule set, if small, is still accurate and general to model the class boundaries sufficiently precisely.

In this paper, we modify and enhance the main components of such system to have improvement in terms of performance measured by test accuracy and the interpretability measured by the size of evolved rule set. Although applying such modification on XCS-HT may result in reducing the evolved rule set, the final

GECCO'11, July 12–16, 2011, Dublin, Ireland.

rule set still has many additional rules which are not only inessential in predicting the correct class label of incoming instance but also in some cases they cause a downfall in the test accuracy. Hence, we employ a reduction method Reduction based on Least Reward Prediction (RLRP) and a compatible inference methodology for the efficient compaction of the final problem solution. This reduction method can eliminate more than 94% of the evolved classifiers on average while hardly affecting the accuracy. It was shown that as compared to XCSR, XCS-HT has fine ability to model the indistinguishable class boundaries. The last but not least motivation of this paper is to assess the effectiveness of the proposed modification of XCS-HT on solving the real world problems. So, several standard data sets are selected from UCI repository database [6] and the obtained results are promising. The results show that our method presents statistically significant improvement in terms of the performance and the number of rules.

The rest of this paper is organized as follows; Section 2 summarizes the important works that have been done to enhance the knowledge representation component of XCS. Section 3 provides a short description of XCS-HT. Section 4 details the proposed enhancement mechanisms can be applied to XCS-HT* in terms of performance and the size of evolved rule set. Section 5 compares our learner with the primary version of XCS-HT, XCSR and Fuzzy-UCS in terms of performance and interpretability measured by the size of evolved rule set. This comparison is done over a collection of real world problems which are selected from UCI repository database [6] to highlight the effectiveness of our method in real world problems with different characteristics. Finally, Section 6 concludes and discusses further work.

2. RELATED WORK

The first and most commonly syntax for classifier condition is *"Ternary"* representation including {0,1,#} alphabets. For handling continuous-valued input, this traditional representation has been substituted to the interval-based representation, proposing an interval in condition part of classifier for each dimension. According to definition of these intervals four different representation techniques have been introduced; Center Spread Representation (CSR) [7], Lower-Upper Bound Representation or Min-Max Representation (MMR) [8], Unordered-Bound Representation (UBR) [9], and Min-Percentage Representation (MPR) [10]. The subspace of problem space that can be presented by a classifier is identified by the disjunction of intervals in condition part of the classifier. Besides, to handle continuous-valued inputs, several valuable efforts have been done to cover the problem space with geometric shape which is defined in condition part of classifiers. Some of these researches are done in [11] and [12] where each classifier presents a hyper ellipsoidal and a convex hull respectively. Furthermore, to make XCS more effective in complex problems, other general purposed representations have been proposed; such as, fuzzy [13-18], GP-like conditions [19], and neural network [20].

There are a number of approaches to use fuzzy logic as a technique to represent fuzzy rules in Michigan style LCS [13-18]. The main motivation behind such efforts is to achieve an online learning system with more accurate, general and well understandable rule set by combining the generalization capabilities of XCS with the fine interpretability of fuzzy rules. Bonarini et al. in [14, 15] introduced the general framework of LCS using fuzzy logic. One of the successful proposed LCS inheriting this framework is Fuzzy-XCS [16] where the rules are expressed in fuzzy format. Soon later, this approach was expanded to be employable in UCS [21] and a system named Fuzzy-UCS was introduced [17, 18]. Fuzzy-UCS tries to combine the good capacity of UCS such as generalization with the good interpretability of fuzzy rules to evolve a compact and understandable rule set. Recently, XCS-HT [5] has been introduced to reduce the size of rules set and maintain the performance as much as possible using hyper-trapezoidal membership function in condition part of classifier.

3. DESCRIPTION OF XCS-HT

This section provides an overall description of the modification of XCS's components in XCS-HT.

3.1 Representation

In XCS-HT, the condition part of each classifier presents two hyper rectangles. One of these hyper rectangles embraces the other one completely. The inner hyper rectangle is indeed the certain region and the space between these two hyper rectangles is a vague region where instances can be matched by less than 100% degree of certainty. An asymmetric hyper trapezoidal membership function is defined in the condition part of a classifier. Thus, there are four genes for each dimension to present such hyper trapezoidal membership function, that is $cl.C=\{a_1,b_1,c_1,d_1,a_2,b_2, c_2,d_2,\ldots, a_n,b_n,c_n,d_n\}$ where n identifies the dimensions of given problem and $\{a_i,b_i,c_i,d_i\}$ are the parameters of the defined trapezoidal membership function in i'th dimension. For better understanding, Figure 1 shows the structure of a classifier in a two dimensional problem and how such classifier partitions the problem space.

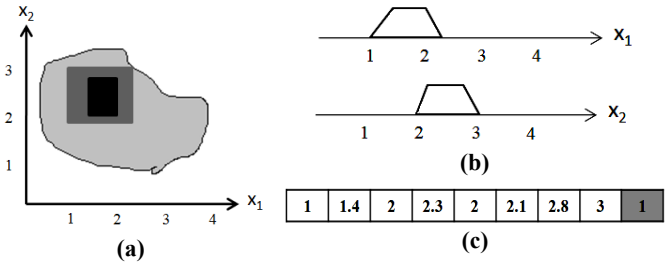

Figure 1. An example of hyper trapezoidal based classifier in an arbitrary problem space. a) Visualization of the area covered by such classifier, including certain region shown in black and vague region, shown in dark gray. b) The trapezoidal membership functions defined for each dimension. c) The genotype of the classifier presented in (a) and (b).

3.2 Matching and Covering

Each classifier has a parameter named matching degree parameter ($cl.\mu$) that shows the degree of certainty in matching the incoming instance. To compute such parameter, a T-norm, like *product(a,b)* or *min(a,b)*, is used to find the conjunction of all matching degrees obtained from the trapezoidal membership function defined in condition part of classifier for each dimension. In our implementation, we use *product(a,b)* as T-norm.

Two methods have been proposed to form [M] named *Ordinary mechanism* and *Roulette Wheel (RW) based mechanism*. In *Ordinary mechanism*, each classifier has a chance to be a member of [M] with a probability which is equal to the value of its matching degree parameter. But, in *RW mechanism*, classifiers which match the incoming instance certainly would be a member of [M] and the other classifiers that match the incoming instance

with $cl.\mu(e)<1$ degree of certainty participate in roulette wheel selection method if the sum of their matching degrees are greater than 0.5.

The covering procedure will be triggered if there is no classifier founded by *RW based mechanism* to match the current instance, the system would generate a sequence of random numbers as $a_1,b_1,c_1,d_1,a_2,b_2,c_2,d_2,\ldots,a_n,b_n,c_n,d_n$ in condition part of new classifier as if it can match the incoming instance certainly.

3.3 Subsumption

Subsumption is a perspicacious technique in XCS to stress additional generalization pressure in that a syntactically more general classifier observed more specialized classifiers [22]. So, it can be concluded that subsumption is a crucial factor along the way to evolve maximally general classifiers. To achieve this goal, we must determine whether the condition part of a classifier $cl_1.C$ is subsumed by the condition part of another classifier $cl_2.C$ which is both accurate ($cl_2.\varepsilon<\varepsilon_0$) and sufficiently experienced ($cl_2.exp>\theta_{sub}$) or not. It is worth mentioning that since subsumption applied in [A], the action part of both classifier cl_1 and cl_2 are the same.

In XCS-HT, to find whether $cl_1.C$ can be subsumed by $cl_2.C$ or not, the overlapping area ($S_{overlap}$) between the two trapezoidal membership functions defined in the condition part of each classifiers is calculated. If in all dimensions the calculated $S_{overlap}$ is greater than a predefined percentage of total area (S_{cl1}) of the trapezoidal membership function of the specified classifier cl_1, ($S_{overlap}>\theta_{overlap}\times S_{cl1}$), cl_1 can be subsumed by the more general classifier cl_2.

4. PROPOSED METHOD

This paper aims at improving the learning mechanism of XCS-HT in terms of test accuracy and also proposing a novel inference mechanism and consequently a compaction method to reduce the final evolved rule set while changing the performance only marginally.

Our proposed method has the same knowledge representation technique that is using hyper trapezoidal membership function to partition the problem space with two kinds of regions. Thus, Each classifier has a parameter named matching degree parameter $cl.\mu(e)$ to identify its matching degree when facing the current incoming state e. Since this parameter has an essential role in evolving better and more compact rule set, we modified the other components of our system to improve its performance. As the reported results in [5] suggested, in our modification the *RW based mechanism* is chosen to form [M] due to its effectiveness in applying more selection pressure than *Ordinary mechanism* and consequently producing less number of rules.

In the following, we describe the modification of XCS-HT's in detail.

4.1 Subsumption

As the knowledge representation used in our system is the same as ones in XCS-HT, we also borrow its subsumption mechanism and in our implementation, we set $\theta_{overlap}$ parameter to 80%.

Figure 2.a illustrates an example of such subsumption mechanism. In this example cl_2 cannot subsume cl_3. On the contrary cl_1 can be subsumed by cl_2 because the overlapping area between cl_1 and cl_2 is by 80% of total area covered by the trapezoidal membership function defined in cl_1

As shown in Figure 2, in some cases when the more general classifier cl_2 subsumes the more specialized one cl_1, some portion of problem space which is covered by the specialized classifier cl_1 would be uncovered. In our example, the area between $cl_2.a$ and $cl_1.a$ will be uncovered after observing cl_1 in subsumption procedure. On the other hand, since the subsumption occurs in training phase, if the specialized classifier is both accurate ($\varepsilon<\varepsilon_0$) and sufficiently experienced ($exp>\theta_{sub}$), subsuming such classifier might cause an unwelcome disregard about some useful and significant knowledge which is obtained through learning process. Hereby, we suggest a merge mechanism which, if simple, is useful to overcome this shortcoming. In subsumption mechanism when an accurate and sufficiently experienced classifier is founded to be subsumed, the parameters of trapezoidal membership function presented in the condition part of the more general classifier, namely the subsumer, will change in that the more specialized classifier is completely subsumed by subsumer classifier. In other words, as shown in Figure 2.b, by applying such merge mechanism the $cl_2.a$ parameter changes and its value would be equal to the value of $cl_1.a$ because cl_2 is a valid subsumer and cl_1 is both accurate and sufficiently experienced. Algorithm 1 sums up the modification of the parameters of trapezoidal membership function presented in the condition part of subsumer.

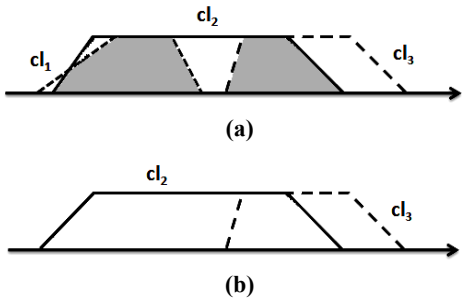

Figure 2. An example of how subsumption mechanism works with $\theta_{overlap}=0.8$. a) cl_2 can subsume cl_1 ($S_{overlap}>\theta_{overlap}\times S_{cl1}$) but cl_2 cannot subsume cl_3 because their membership functions are not overlapped enough ($S_{overlap}>\theta_{overlap}\times S_{cl3}$). b) The suggested merge mechanism. Since cl_1 can be subsumed by cl_2, the cl_2's membership function can be extended as if cl_1 were completely subsumed by cl_2 ($cl_2.a$=min($cl_1.a$,$cl_2.a$) and $cl_2.b$=min($cl_1.b$, $cl_2.b$)).

Algorithm 1: Subsumption operator with merge mechanism

```
initialize subsumer
for each classifier cl in [A]
    subsumer = Empty
    if (cl.exp>θsub && cl.ε<ε0)
        subsumer ← cl
    if (subsumer is not Empty)
        for each cl in [A]
            if (IS MORE GENERAL (subsumer , cl ) )
                subsumer.num ← subsumer.num + cl.num
                if (cl.exp>θsub && cl.ε<ε0)
                    for each dimension i
                        subsumer.ai ← min(subsumer.ai , cl.ai )
                        subsumer.bi ← min(subsumer.bi , cl.bi )
                        subsumer.ci ← max(subsumer.ci , cl.ci )
                        subsumer.di ← max(subsumer.di , cl.di )
                remove classifier cl form set [A]
                remove classifier cl form set [P]
```

4.2 Rule Evaluation

In each extension of XCS such as our system, each rule or namely classifier has following main parameters and several addition estimates.

1. Reward prediction error (*cl.ε*): estimates the mean absolute deviation of *cl.P* with respect to the actual reward (*R*).
2. Reward prediction (*cl.P*): estimates the average reward received if the *cl.C* is satisfied and *cl.A* is executed.
3. Fitness (*cl.F*): estimates the scaled, relative accuracy of *cl* regarding to other classifier being in [A].
4. Experience (*cl.exp*): counts the number of times that *cl* has belonged to an action set [A].

The parameters of all classifiers in [A] would be updated with respect to the successive problem instance along exploration phase or training phase.

In our system, each classifier has an additional parameter named matching degree *cl.μ*. This parameter identifies that the underlying classifier on what certainty degree can match the incoming instance. Here, we use this information to update the corresponding parameters of classifiers. In other words, at the end of each learning iteration, the parameter of all the classifiers belonging to [A] are updated with respect to their matching degree with the input instance *e*.

The first parameter that must be updated is reward prediction error. In XCS-HT, we update it as follows:

$$\text{cl.}\varepsilon \leftarrow \text{cl.}\varepsilon + \beta\,(\,|R - \text{cl.P}| \times \text{cl.}\mu(e) - \text{cl.}\varepsilon\,) \tag{1}$$

And the second parameter is reward prediction that would be updated as follows:

$$\text{cl.P} \leftarrow \text{cl.P} + \beta\,(\,R - \text{cl.P}\,) \times \text{cl.}\mu(e) \tag{2}$$

It is worth mentioning that the condition part of each classifier presents two kinds of regions, namely certain and vague region, by defining a hyper trapezoidal membership function. Hereby, for better modeling the regularities of environment, the parameters of each classifier must be updated in view of the fact that the certain region would be as accurate as possible whereas it is not necessary that the vague region is evolved as accurately as is the certain region. Such being the case, we suggest that the reward prediction error and the prediction error of each classifier are updated with respect to the matching degree parameter as Equation 1 and 2 show. If the incoming instance *e* is matched by the certain part of a classifier *cl*, Equation 1 and 2 will be similar to the one in XCS since the matching degree of the corresponding classifier is equal to one (*cl.μ*(*e*)=1). On the other hand, if *e* matches by the vague region of *cl* with *cl.μ*(*e*) < 1 degree of certainty, it is expected that the receiving reward *R* affects the estimation of reward prediction error and prediction error less, so we multiply (*R* – *cl.P*) by *cl.μ*(*e*).

The fitness parameter is updated as suggested in XCS. Equation 3, 4 and 5 provide a recollection of how the fitness parameter would be updated. Since for calculating the accuracy of each classifier the current value of its reward prediction error is used and we update *cl.ε* according to *cl.μ*(*e*), the updated value of fitness parameter *cl.F* is implicitly affected by the matching degree parameter of each classifier.

$$\text{cl.k} = \begin{cases} 1 & \text{if } \text{cl.}\varepsilon < \varepsilon_0 \\ \alpha\left(\dfrac{\text{cl.}\varepsilon}{\varepsilon_0}\right)^{-\nu} & \text{otherwise} \end{cases} \tag{3}$$

$$\text{cl.k}' = \frac{\text{cl.k} \times \text{cl.num}}{\sum_{\text{cl}\in[\text{A}]} \text{cl.k} \times \text{cl.num}} \tag{4}$$

$$\text{cl.F} \leftarrow \text{cl.F} + \beta\,(\,\text{cl.k}' - \text{cl.F}\,) \tag{5}$$

The experience parameter of each classifier (*cl.exp*) counts the number of times the condition part of this classifier is satisfied and its action is applied. In other words, *cl.exp* shows number of times that *cl* belongs to [A]. In XCS-HT, as the condition part of each classifier is satisfied by *cl.μ*(*e*) degree of certainty, the experience parameter *cl.exp* is updated as follows:

$$\text{cl.exp} \leftarrow \text{cl.exp} + \text{cl.}\mu(\text{e}) \tag{6}$$

4.3 Class Inference Methodology

Like any reinforcement learning system, XCS works in two different modes: exploration or training and exploitation or test. XCS is an online learning system interacting with environment along exploration mode to evolve a maximally general rule set containing rules with minimum prediction error. In exploitation mode, XCS uses such evolved rule set to predict the best action or class label in classification tasks for new incoming instance.

To select the current action, XCS firstly computes the system prediction $P(a_i)$ for each possible action using $P(a_i) = \dfrac{\sum_{cl\in[M]\wedge cl.A=a_i} cl.F \times cl.P}{\sum_{cl\in[M]\wedge cl.A=a_i} cl.F}$. Then, the winner action is chosen depending on the policy to explore or exploit. $P(a_i)$ shows the fitness weighted average of the reward prediction estimates of the classifier in [M] that advocates action a_i.

Here, we use a similar strategy to select the winner action by calculating $P(a_i)$. The consequent system prediction entries are computed as follows:

$$P(a_i) = \frac{\sum_{cl\in[M]\wedge cl.A=a_i} cl.\mu(e) \times cl.F \times cl.P}{\sum_{cl\in[M]\wedge cl.A=a_i} cl.\mu(e) \times cl.F} \tag{7}$$

As the incoming instance *e* is not matched with all classifiers in [M] with same certainty degree, so it is fair to compute the $P(a_i)$ by considering the matching degree of each classifiers in [M]. In Equation 7, the fitness of each classifier is multiplied by the value of its matching degree parameter to reduce the influence of the reward prediction of classifiers that the vague region presented in their condition part matches the new instance.

4.4 Rule Set Reduction

XCS is a population based learning system and consequently the evolved classifier population codes the final solution to the given problem. The problem solution is overrepresented with similar, multiple, strongly overlapping classifiers. Thus, an important part of LCS-based system is effective rule set compaction. Here, we employ a compaction method termed Reduction based on Least Reward Prediction (RLRP). RLRP can reduce the evolved

population by 94% on average while hardly affecting the test accuracy.

The proposed reduction method tries to minimize the rule set size by selecting the more general classifiers which have not only more reward prediction estimate *cl.P* but also less reward prediction error *cl.ε*. In other words, RLRP tries to find classifiers that maximize the least reward prediction which is estimated by (*cl.P-cl.ε*) with respect to their generality namely *cl.generality*. The pseudo code of this reduction method is provided in Algorithm 2.

Algorithm 2: Rule Set Reduction

```
FinalRuleSet=Empty
for each train instance e
    classifier c=FinalRuleSet.getmatchingClassifier(e)
    if c==null  c.A!=e.classLabel
        max=0 , bestCl=null
        for each classifier cl in [P]
            if cl.match(e)  cl.A=e.classLabel
                cmax=(cl.P-cl.ε)×cl.generatilty
                if max<cmax
                    max=cmax
                    bestCl=cl
            FinalRuleSet.add(cl)
```

The proposed compaction method is applied at the end of the learning process to obtain a minimum set of rules. For each training sample, we check the classifier population to find the classifier that maximizes *(cl.P-cl.ε)×cl.generality* and copy such classifier to the final population. *cl.generality* shows how much *cl* is general. Generality is defined as the sum of the widths of certain region and half of the widths of vague region that are presented in the classifier condition part, as follows:

$$\begin{aligned} \text{cl.generality} &= \sum_{\text{dimension } i} (\text{cl.c}_i - \text{cl.b}_i) \\ &\quad + \frac{1}{2} \sum_{\text{dimension } i} ((\text{cl.b}_i - \text{cl.a}_i) + (\text{cl.d}_i - \text{cl.c}_i)) \\ &= \frac{1}{2} \sum_{\text{dimension } i} \text{cl.c}_i - \text{cl.b}_i + \text{cl.d}_i - \text{cl.a}_i \end{aligned} \quad (8)$$

We try to avoid the situation that the problem space would be uncovered by considering *cl.generality* in a criterion that must be maximized for each train instance.

Since the proposed reduction method is applied at the end of learning iteration, we suggest a new inference method in test mode. The proposed inference method is compatible with our reduction method to affect the test accuracy as marginally as possible. Here, to predict the correct action or class label in classification tasks for each incoming instance *e*, the compact rule set is checked to find the winner rule *cl* which maximizes *(cl.P-cl.ε)×cl.μ(e)*. As the classifiers match the new incoming instance *e* with different degrees of certainty, we multiple the least reward that a classifier *cl* predicts to receive by its matching degree *cl.μ*(*e*) that shows how much *cl* is successful in matching the unseen data *e*. In other words, by considering the value of the matching degree of a classifier, the influence of the reward prediction of those classifiers that their vague region matches the new instance is reduced.

5. EXPERIMENTAL RESULTS

To verify the basic behavior of XCS-HT* facing real valued problems, the experiments were carried out selecting eight real world problems with different characteristics from UCI repository [6]: *Balance-scale* (bal), *Glass* (gls), *Iris* (irs), *Thyroid* (thy), *Pima* (pim), *Vehicle* (veh), *Wisconsin diagnose breast cancer* (wbcd), and *Wine* (wne).

The parameter of XCS-HT* is configured as follows: numIter=100000, population size=6400, β=0.2, α=0.1, ν=5, χ=0.8, τ=0.4, μ=0.04, r_0=0.1, m_0=0.6, θ_{GA}=48, θ_{del}=50, θ_{sub}=50, δ=0.1 and $\theta_{overlap}$=80. It is worth mentioning that this parameters setting also used on XCS-HT, Fuzzy-UCS and XCSR according to their authors' suggestions.

We compared the performance, as well as the interpretability of the learners selected to be compared with each other. The performance was measured by the test accuracy rate and the interpretability of learners was quantified by the size of evolved rule set. To have reliable estimations of these indicators, ten-fold cross validation [23] was used.

5.1 Comparison among Different Versions of XCS-HT*

This section furthers the study on the enhancement mechanisms introduced in the proposed modification of XCS-HT which was termed as XCS-HT*. To verify the effectiveness of XCS-HT*, Table 1 provides a comparison between the original version of XCS-HT and different versions of the proposed learner, XCS-HT*, each of which uses one or some enhancement mechanisms introduced in previous section. The learners which are compared to each other in Table 1 are listed as follows:

XCS-HT: (presented in second column) is a simple version of XCS-HT which was firstly introduced in [5].

XCS-HT$_{rr}$: (presented in third column) is a version of XCS-HT in that its rule set is refined by eliminating the classifiers that their experiences *cl.exp* are less than $\theta_{exploit}$, a user-set parameter. Here, we set it to 10 as it was suggested in [18].

XCS-HT*(RE)$_{rr}$: (presented in fourth column) Here, the subsumption mechanism is modified as introduced in Section 4.1 and the rule evaluation mechanism proposed in Section 4.2 is used to improve the performance of system.

XCS-HT*(RE+IM)$_{rr}$: (presented in fifth column) likes the previous one. In addition, its class inference methodology is modified based on the matching degree of each classifier as suggested in Section 4.3.

XCS-HT*(RE+RLRP)$_{rr}$: (presented in sixth column) in addition to updating the parameters of classifiers according to equations proposed in Section 4.2, after eliminating the classifiers with *cl.exp*<$\theta_{exploit}$, it uses RLRP reduction method and its compatible class inference methodology proposed in 4.4.

Each above learner was applied on all real world problems. Then we examine the trade-off between the obtained performance and the size of model created by each learner. The two last rows of Table 1 indicate the average of the rank over independent runs on selected UCI data sets and the position of each learner in the ranking in terms of obtained performance and the rule set size respectively.

Table 1: The test accuracy and the number of rules of the models created by different versions of XCS-HT* and XCS-HT.

	Performance					Number of rules				
	XCS-HT	XCS-HT$_{rr}$	XCS-HT* (RE)$_{rr}$	XCS-HT* (RE+IM)$_{rr}$	XCS-HT* (RE+RLRP)$_{rr}$	XCS-HT	XCS-HT$_{rr}$	XCS-HT* (RE)$_{rr}$	XCS-HT* (RE+IM)$_{rr}$	XCS-HT* (RE+RLRP)$_{rr}$
bal	84.68%	84.26%	**85.43%**	85.22%	84.19%	1558	604	642	642	**42**
gls	**70.77%**	68.80%	66.67%	66.51%	64.13%	2964	1125	1170	1170	**37**
irs	94.94%	94.48%	95.11%	**95.78%**	94.89%	96	70	77	77	**8**
thy	92.93%	93.27%	94.60%	**94.76%**	93.96%	373	176	196	196	**13**
pim	74.18%	74.14%	75.31%	**75.75%**	73.77%	3675	1548	1599	1599	**98**
veh	72.49%	72.17%	**72.94%**	72.26%	67.67%	4814	1684	1780	1780	**91**
wdbc	96.00%	95.87%	95.77%	**96.04%**	93.04%	3992	1835	1825	1825	**52**
wne	**94.73%**	94.32%	94.31%	94.31%	89.02%	2870	1852	1866	1866	**21**
rank	2.55	3.35	2.5	**2.1**	4.5	4	2.1	2.9	2.9	**1**
pos	3	4	2	**1**	5	4	2	3	3	**1**

Table 2: Comparison of XCS-HT* and XCS-HT*(RLRP) with XCSR, primary version of XCS-HT and three versions of FuzzyUCS in terms of performance and the size of evolved rule set.

	XCSR		XCS-HT		XCS-HT*		XCS-HT* (RLRP)		FuzzyUCS (wavg)		FuzzyUCS (awin)		FuzzyUCS (nfit)	
	per	*pop*	*per*	*pop*	*per*	*pop*	*per*	*pop*	*per*	*pop*	*per*	*pop*	*Per*	*pop*
bal	82.65%	1860	84.68%	1558	85.22%	642	84.19%	**42**	**88.56%**	1212	84.40%	114	83.40%	75
gls	**71.52%**	3394	70.77%	2964	66.51%	1170	64.13%	37	60.65%	2799	57.21%	62	57.43%	**36**
irs	94.69%	724	94.94%	96	**95.78%**	77	94.89%	8	95.67%	480	95.47%	18	93.73%	**7**
thy	**95.72%**	1679	92.93%	373	94.76%	196	93.96%	**13**	88.18%	3130	89.49%	138	91.25%	28
pim	73.23%	3320	74.18%	3675	**75.75%**	1599	73.77%	98	74.88%	2841	74.11%	192	74.32%	**62**
veh	**72.84%**	4830	72.49%	4814	72.26%	1780	67.67%	**91**	67.68%	3732	65.35%	332	65.34%	147
wdbc	94.97%	5324	96.00%	3992	**96.04%**	1825	93.04%	**52**	95.20%	5412	94.61%	276	94.51%	101
wne	**97.00%**	4086	94.73%	2870	94.31%	1866	89.02%	**21**	94.12%	3686	94.86%	95	91.82%	26
rank	3.5	6.5	2.95	5.5	**2.18**	4	5.31	**1.25**	3.4	6	4.69	3	5.69	1.75
pos	4	7	2	5	**1**	4	6	**1**	3	6	5	3	7	2

Table 3: Pairwise comparison of the performance (white cells) and the size of evolved rule set (gray cells) of XCSR, XCS-HT, XCS-HT*, XCS-HT*(RLRP), three versions of FuzzyUCS by means of Wilcoxon signed-ranks test (α=0.05).

	XCSR	XCS-HT	XCS-HT*	XCS-HT* (RLRP)	FuzzyUCS (wavg)	FuzzyUCS (awin)	FuzzyUCS (nfit)
XCSR		0.8438 □	0.9453 □	0.1094 ○	0.4609 □	0.2500 ○	0.1094 ○
XCS-HT	0.0391 ■		0.5469 □	0.0547 ○	0.3125 ○	0.1094 ○	0.0156 ●
XCS-HT*	0.0078 ■	0.0078 ■		0.0078 ●	0.0781 ○	0.0234 ●	0.0078 ●
XCS-HT* (RLRP)	0.0078 ■	0.0078 ■	0.0078 ■		0.5469 □	0.9453 □	0.4609 ○
FuzzyUCS (wavg)	0.2500 □	0.6406 ○	0.0078 ●	0.0078 ●		0.1953 ○	0.1094 ○
FuzzyUCS (awin)	0.0078 ■	0.0078 ■	0.0078 ■	0.0078 ●	0.0078 ■		0.6406 ○
FuzzyUCS (nfit)	0.0078 ■	0.0078 ■	0.0078 ■	0.2344 ○	0.0078 ■	0.0078 ■	

As shown in Table 1, eliminating classifiers with $cl.exp<\theta_{exploit}$ (XCS-HT$_{rr}$) decreased the size of rule set over 51.7% on average. Nonetheless, it went against the performance which was fallen down. To achieve better position in the performance ranking, in Section 4 we suggested that the parameters of classifier would be updated with respect to their certainty degree in matching of incoming instance and a similar strategy was introduced to infer the class of new example. The obtained results are promising since XCS-HT*(RE+IM)$_{rr}$ got the best rank in the presented comparison. On the other hand, our reduction method, namely RLRP, helped XCS-HT* for considerable reduction of the rule set while slightly degrading the performance. As shown in Table 1, RLRP was able to boil the number of rules created by XCS-HT* down by often more than 94% while only decreasing the performance about 2%. So, it is obvious that there exists a trade-off named performance- interpretability trade-off between the test accuracy as a performance measurement and the size of the model created by XCS-HT* as an interpretability measurement.

According to the performance-interpretability trade-off, XCS-HT*(RE+IM)$_{rr}$ is the best ranked in the comparison of performance and XCS-HT*(RE+RLRP)$_{rr}$ considerably improves the interpretability due to its large reduction of the rule set. To have fair comparison with other learners such as XCSR, XCS-HT and three versions of Fuzzy-UCS, we selected these two versions of the proposed system and termed them XCS-HT* and XCS-HT*(RLRP) respectively. XCS-HT* is indeed XCS-HT*(RE+IM)$_{rr}$ which was improved by enhancement mechanisms proposed in Section 4 and have the best rank in terms of performance. Similarly, XCS-HT*(RLRP) is the one using the proposed reduction method and being the best ranked among others in terms of number of rules as it was shown in Table 1.

5.2 Comparison with other learners

Table 2 detailed a comparison of the proposed learners, XCS-HT* and XCS-HT*(RLRP), with the most common Michigan style learner in the literature called XCSR using interval based representation to handle real valued problems [7,8], the original version of XCS-HT which is proposed in [5] and three versions of Fuzzy-UCS [17,18], a Michigan style learning classifier system, introducing a linguistic representation of the rules and reduction methods with the aim of reducing the size of evolved rule set while maintaining similar performance. This comparison was done in terms of performance measured by the test accuracy rate and interpretability measured by the number of rules of the models created by these learners. It is worth mentioning that the results of Fuzzy-UCS presented in Table 2 equal to the values that have been reported in [18].

As this table shows, the obtained result is encouraging. XCS-HT* is able to reach the maximum test accuracy among all learners. It indicates that XCS-HT* have better ability to model the problem regularities and surpassed the performance of XCS-HT and XCSR, respectively the second and third best ranked learners. Moreover, XCS-HT* results in better rank in comparison with these two algorithms in terms of the number of rules evolved by the underlying learner (see the *pop* column of the last row of Table 2). With regard to the interpretability of the learners measured by the number of rules created by the learner, the fine performances of XCSR and Fuzzy-UCS with weighted average [18], Fuzzy-UCS(wavg), were hampered by evolving the biggest and the second biggest rule set. On the other hand, XCS-HT*(RLRP) is the best algorithm and Fuzzy-UCS with most numerous and fittest rules [18], Fuzzy-UCS(nfit), is the second one in ranking. They allowed for considerable reduction of the rule set while slightly degrading the performance.

By comparing our method, XCS-HT*, to the old version of XCS-HT, it can be concluded that XCS-HT* can achieve better performance with so smaller evolved rule set. With respect to Table 2, XCS-HT* is the best ranked algorithm whereas the size of its rule set is about half of the size of the rule set created by the second best ranked algorithm in terms of performance (XCS-HT). Consequently, it shows that enhancement mechanisms proposed in Section 4 are effective to improve XCS-HT* in terms of both performance and the number of evolved rules. Moreover, the results of XCS-HT*(RLRP) in Table 2 show that with the proposed reduction method (RLRP), it is possible to improve the interpretability considerably since it reduces the size of the models created by XCS-HT* over than 94% on average while its performance decreases only about 2%.

In comparison with XCSR, XCS-HT* have better performance with producing only 33% of the rules created by XSCR. In XCS-HT*, each classifier presents two kinds of regions, namely certain and vague region, whereas in XCSR each classifier only present the certain one. Hereby, it is expected that XCS-HT* be able to evolve more general classifiers while their certain region are as accurate as possible and their vague region are as accurate as needed. Compared to three versions of Fuzzy-UCS, Table 2 shows that our system, XCS-HT*, achieved better performance with smaller rule set than the Fuzzy-UCS(wavg) produced. Nevertheless, its rule set size was still extremely larger than two other versions of Fuzzy-UCS with *awin* and *nfit* inferences. However, the results of applying RLRP and its compatible inference method are optimistic. As Table 2 shows, RLRP can pave the way for decreasing the size of models created by XCS-HT* substantially while the performance decreases marginally. In comparison to Fuzzy-UCS(nfit), a version of Fuzzy-UCS which can evolve rule sets that ranged from tens to few hundreds of rules, XCS-HT*(RLRP) is able to achieve a little better performance while evolving less number of rules.

Tables 3 illustrated the p-values resulting to a non-parametric wilcoxon signed-ranks test [24] at α=0.05. The p-values obtained by comparing the test performance are in upper triangular cells which are in white. The lower triangular cells colored in gray indicate the p-values of comparing the size of rule set created by selected learners. The ● and ■ symbols denote that the method in the row significantly improved/degraded the performance or the size of rule set obtained with the learner in the column. The ○ and □ indicate a non-significant improvement/degradation. The results highlight the high competitiveness of XCS-HT* in terms of performance and interpretability with respect to other selected learners. In addition, as expected RLRP can significantly degrade the size of model created by XCS-HT* and also provide a compatible performance.

6. CONCLUSIONS AND FUTURE WORK

XCS-HT* is an extension of XCS-HT, a Michigan style learning classifier system introducing a fuzzy approach to represent knowledge. Each classifier can partition the problem space with two kinds of regions named certain and vague regions. This means that each classifier can match an incoming instance *e* with a degree of certainty that is $0 \leq cl.\mu(e) \leq 1$. This paper has attempted to improve this approach and assess the effectiveness of our method named XCS-HT* on real world problems. Hence,

some essential components of XCS-HT* such as subsumption mechanism, rule evaluation, class inference methodology have been enhanced to improve its potency and performance in dealing with problems especially real world problems. All these suggested enhancement mechanisms are based on how much a classifier is successful to match the new unseen data *e*, namely matching degree parameter $cl.\mu(e)$.

The main aim of this paper is not only to improve XCS-HT* in terms of performance as much as possible but also to enhance its interpretability by reducing the size of its rule set as less as possible. Therefore, a novel reduction method named RLRP has been proposed. It is based on maximizing the least reward prediction estimated by selected classifiers. Additionally, a new inference method which is compatible with RLRP has been suggested to maintain the similar performance. The performance and the size of the models created by XCS-HT* have been tested on a collection of real world problems selected from UCI repository. The obtained results identified that XCS-HT* tries to evolve a promising rule set consisting more general classifiers while their certain region are as accurate as possible and their vague region are as accurate as needed. This research opens up many research directions; first handling mixed attributed problems and second improving XCS-HT* to be applicable on multistep problems with continuous inputs.

7. REFERENCES

[1] J. H. Holland. Adaptation. In R. Rosen and F. Snell, editors, *Progress in Theoretical Biology*, volume 4, pages 263–293. New York: Academic Press, 1976.

[2] S.W. Wilson. Classifier fitness based on accuracy. *Evolutionary computation*, 3(2):149-175, 1995.

[3] P.L. Lanzi. Learning classifier systems: then and now. *Evolutionary Intelligence*. 1(1): 63–82, 2008.

[4] T. Kovacs. XCS classifier system reliably evolves accurate, complete, and minimal representations for boolean functions. In Roy, Chawdhry, and Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 59–68, 1997.

[5] F. Shoeleh, A. Hamzeh and S. Hashemi. To Handle Real Valued Input in XCS: Using Fuzzy Hyper-trapezoidal Membership in Classifier Condition. *Simulated Evolution and Learning*, pages 55-64, 2010.

[6] A. Asuncion and D. J. Newman. *UCI Machine Learning Repository: [http://www.ics.uci.edu/~mlearn /MLRepository.html]*. University of California, 2007.

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

[8] S. W. Wilson. Mining oblique data with XCS. *Advances in Learning Classifier Systems*, pages 158–176, 2001.

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

[10] H. H. Dam, H. A. Abbass, and C. Lokan. Be real! XCS with continuous-valued inputs. In *GECCO'05: In Proceedings of the 2005 Genetic and Evolutionary Computation Conference workshop program*, pages 85–87, 2005.

[11] M. V. Butz, P. L. Lanzi, and S. W. Wilson. Function approximation with XCS: Hyperellipsoidal conditions, recursive least squares, and compaction. *IEEE Transactions on Evolutionary Computation*, 12(3):355–376, 2008.

[12] P. L. Lanzi and S. W. Wilson. Using convex hulls to represent classifier conditions. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1481–1488, 2006.

[13] M. Valenzuela-Rend´on. The fuzzy classifier system: A classifier system for continuously varying variables. In *4th ICGA*, pages 346–353. Morgan Kaufmann, 1991.

[14] A. Bonarini and C. Bonacina and M. Matteucci. Fuzzy and crisp representations of real-valued input for learning classifier systems. *Learning Classifier Systems*, pages 107-124, 2000.

[15] A. Bonarini. An introduction to learning fuzzy classifier systems. *Learning Classifier Systems*, pages 83-104, 2000.

[16] J. Casillas, B. Carse, and L. Bull. Fuzzy-XCS: A Michigan genetic fuzzy system. *IEEE Transactions on Fuzzy Systems*, 15(4):536–550, 2007.

[17] A. Orriols-Puig, J. Casillas, and E. Bernad´o-Mansilla. Fuzzy-UCS: Preliminary results. In *GECCO'07: Proceedings of the 2007 Genetic and Evolutionary Computation Conference Workshop Program*, volume 3, pages 2871–2874, 2007.

[18] A. Orriols-Puig, J. Casillas, and E. Bernad´o-Mansilla. Fuzzy-UCS: a Michigan-style learning fuzzy-classifier system for supervised learning. *IEEE Transactions on Evolutionary Computation*, 13(2): 260-283, 2009.

[19] S. W. Wilson. Classifier conditions using gene expression programming. Technical report, IlliGAL Report No. 2008001, Urbana-Champaign IL 61801, USA, 2008.

[20] L. Bull and T. O'Hara. Accuracy-based neuro and neuro-fuzzy classifier systems. In *GECCO'02: Proceedings of the 2002 Genetic and Evolutionary Computation Conference*, pages 905–911, 2002.

[21] E. Bernad´o-Mansilla and J. M. Garrell. Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

[22] M. V. Butz. *Rule-based evolutionary online learning systems: A principled approach to LCS analysis and design*, volume 109 of *Studies in Fuzziness and Soft Computing*. Springer, 2006.

[23] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

[24] F. Wilcoxon. Individual comparisons by ranking methods. Biometrics, 1:80–83, 1945.