Analysis of Reduction Algorithms for XCS Classifier System

Albert Orriols i Puig
Ester Bernadó i Mansilla
Enginyeria i Arquitectura La Salle. Universitat Ramon Llull
Passeig Bonanova, 8. 08022 Barcelona
{*aorriols,esterb*}*@salleURL.edu*

Abstract. XCS is a classifier system that combines reinforcement learning and genetic algorithms to learn a set of rules describing the knowledge inherent in a dataset. Recent studies have shown that XCS is highly competitive with respect to other classifier schemes. However, these studies have been mainly based on the analysis and improvement of the classification accuracy, paying few attention to the explanatory ability of the system. This paper focuses on the latter aspect. We highlight the high number of rules that XCS evolves in real-world datasets, which makes the human interpretation of the extracted knowledge difficult. Therefore, we investigate several algorithms that reduce the ruleset after training, with the aim of obtaining a simpler explanation of the dataset without loosing much generalization ability. We compare three algorithms in terms of the number of rules that are finally obtained, the loss in classification accuracy and the CPU cost. We also find that training XCS during high number of iterations does not improve the classification accuracy of the system but increases the reliability of the reduction algorithms.

Key Words: Evolutionary Computation, Classifier Systems, Machine Learning, Data Mining, Classification.

1 Introduction

XCS [10, 11] is a classifier system that is recently arising as a promising method for classification problems and data mining. It has shown to be highly competitive with respect to other machine learning techniques, such as nearest neighbor algorithms, decision trees, decision forests, etc [2, 1], both in terms of generalization and explanatory capabilities.

In data mining, the explanatory capability of a classifier is as important as its generalization ability. In fact, in many real-world domains (e.g, medical, financial), human experts are in search of knowledge extracted from the databases, rather than simply a model that can predict accurately new cases but without accompanying explanations. In this framework, XCS can offer good accuracy results together with an explanation based on a set of rules.

Some studies have shown that XCS tends to evolve a compact ruleset, that is, a ruleset formed by a minimum number of general rules [7]. But although this is true for classification problems with binary and nominal attributes, this does not happen with real-world problems, where attributes are usually real numbers. In these types of problems, XCS uses a hyper-rectangle representation [12, 9] that tends to produce a high number of rules. This implies

little explanatory capability, i.e., more difficulty to interpret the knowledge representation extracted in the ruleset. Moreover, the evolution of high number of rules may produce higher training times.

A possible solution to this problem would be to use a discretization of attributes and allow XCS to work as if they were nominal values. But his would impose a limit on the generalization ability that the system could achieve, since the discretization would fix a priori the set of intervals that should be used. Instead, we prefer to allow XCS to evolve these intervals automatically and apply a posteriori process to reduce the ruleset. In fact, most of the rules that are evolved with the hyperrectangle representation are redundant rules, which overlap partially with others and are not necessary to achieve a good classification accuracy. The goal of the reduction algorithms is to eliminate such rules without loosing much classification accuracy.

This paper analyses different proposals of reduction algorithms in several real-world classification problems. We study some algorithms proposed in the literature [13, 6], which are not compared deeply so that it is not clear which algorithm offers better benefits. We analyze these algorithms in terms of the generalization ability after the reduction algorithm, the number of rules obtained and the computational cost. The paper is structured as follows. Section 2 gives a brief description of XCS and its representation for real attributes. Section 3 describes the three reduction algorithms used in our study. Next, we present the experimental set and give a comparative between these algorithms. Finally, we give our conclusions and guidelines for future work.

2 Description of XCS

XCS is a classifier system that learns iteratively from the interaction with the environment, according to a reinforcement learning scheme, and applies a genetic algorithm as a search component. In the following we give a brief description of XCS.

2.1 Representation

XCS evolves a population [P] of classifiers, where each classifier has a rule that consists of a condition part and a class: $condition \rightarrow class$. For real attributes, XCS uses the hyperrectangle representation, where the condition part is a set of intervals $[l_i, u_i]^l$, where l is the length of the input. Thus, a rule's condition matches an input example $(x_1, x_2, ..., x_l)$, if $\forall i \; l_i \leq x_i \leq u_i$. The action of the rule is codified as an integer.

To estimate the quality of each rule, there are three main parameters: a) the payoff prediction p, an estimate of the payoff that the classifier will receive if its class is selected, b) the prediction error ϵ, which estimates the error between the classifier's prediction and the payoff and c) the fitness F, computed as an inverse function of the prediction error.

2.2 Performance Component

XCS learns incrementally, according to a reinforcement learning scheme. Each time step, an input example x coming from the training set is presented to the system. Then, XCS builds a match set [M], consisting of those classifiers whose conditions are satisfied by the example. Then, a classification is chosen from those available in [M]. This classification can be selected from a variety of regimes, ranging from the *pure-explore mode* to the *pure-exploit*

mode. In pure-explore mode, the class is selected randomly. In pure-explore mode, we select the most promising class. Under classification, pure-explore is used during training, because the system is learning the consequences of using each of the classes. Pure-exploit is used when the system is predicting new unseen examples. Once the class is selected, XCS builds the action set [A], which is formed by all the classifiers in [M] proposing this class.

2.3 Reinforcement Component

Once the class is selected, the environment returns a reward r, which is used to adjust the parameters (prediction, error and fitness) of the classifiers in [A] (see [4] for the details).

2.4 Discovery Component

The genetic algorithm (GA) in XCS is used as a search mechanism, discovering promising rules from the recombination of existing ones and deleting those rules that do not contribute to the knowledge. The GA is applied to the action sets. It selects two parents from the current action set [A] with probability proportional to fitness. Then, the parents are crossed and mutated with probabilities χ and μ respectively.

The resulting offspring are introduced into the population. First, each offspring is checked for subsumption [11] with its parents. If either of the parents is reliable, accurate and more general than the offspring, then the offspring is not introduced on behalf of its parent. Subsumption tries to eliminate specific classifiers that are covered by more general versions, tending thus to condensate the population. If the offspring classifier can not be subsumed, it is inserted in the population, deleting a potentially poor classifier if the population is full [8].

3 Compact Rulesets: Description of Algorithms

In this section we describe three reduction algorithms. These algorithms are applied once the training is performed with the aim of compacting the final ruleset without loosing accuracy.

3.1 Wilson's Algorithm

The first proposal of a reduction algorithm for XCS appears in [13]. The algorithm works as follows:

1. First, the population of rules [P] is sorted according to some criterion, such as the numerosity of the rules or their experience. This sorted population is denoted as M. Then, the algorithm selects the minimum sequential set of classifiers M_n that achieves the same generalization accuracy as M, when tested over the training set $\mathcal{T}$. M_n is formed by all classifiers c_i, where $i \leq n$.

2. Once M_n is obtained, the algorithm deletes those classifiers that do not increase the classification accuracy. That is, if the accuracy of M_i is less or equal than the accuracy of M_{i-1}, then classifier c_i is deleted. The resulting population is denoted as B.

3. Next, the algorithm tries to delete the redundant classifiers. That is, it selects the minimum set of classifiers that cover all the training instances. This is an iterative process. In each

step, it selects the rule that covers the maximum number of instances of $\mathcal{T}$. This rule is copied into the final ruleset C and the instances that it covers are deleted from $\mathcal{T}$. The process continues until $\mathcal{T}$ is empty.

Thus, the first two steps of the algorithm try to select the minimum number of classifiers that achieve the same classification accuracy as the original population. The third step selects the minimum set of classifiers covering all the training samples, where each classifier is selected maximizing coverage of examples.

3.2 Proposals by Dixon et al

In [6] there are several proposals of alternative reduction algorithms. Based on them, we have designed two variants of a algorithm that collects the main ideas proposed by Dixon et al. The algorithm is composed of two main steps:

1. Deletion of non-qualified classifiers. A qualified classified has high experience, low error and high prediction. To be exact, a qualified classifier is considered as one that achieves: $exp > \theta_{reduct}$, $\epsilon \leq \epsilon_{reduct}$ and $p \geq p_{reduct}$, where θ_{reduct}, ϵ_{reduct} and p_{reduct} are thresholds set by the user.

2. Deletion of non-useful classifiers. A useful classifier is considered as one that is used for the classification of some training example, similarly to the third step of Wilson's algorithm. However, the computation is different. Here, for each example of the training set $\mathcal{T}$, we form a match set [M] and we mark as useful those classifiers of the match set that contribute to the classification of the example. To be exact, from the match set [M] we compute the winning class, as explained in section 2.2. Then, we mark as useful only the classifier with the highest prediction among those proposing the winning class. Another possibility is to mark as useful all the classifiers proposing the selected class. We call these versions as D1 and D2 respectively.

4 Comparison of Reduction Algorithms

In this section we compare the three reduction algorithms, named W, D1 and D2 respectively. The comparison is made using an stratified ten-fold cross-validation test [5] on ten datasets containing only real attributes. Most of these datasets belong to the UCI repository [3]: bupa liver disorders (`bpa`), Wisconsin breast cancer (`bre`), glass (`gls`), iris (`irs`), pima indians diabetes (`pmi`), vehicle (`veh`) and wine (`wne`). Biopsies (`bps`), mammographies (`mmg`) and `tao` belong to our own repository (see [2] for the details).

We have run XCS with the following parameter settings (see [4] for the notation): $iter. = 100,000$, $N = 6400$, $\beta = 0.2$, $\theta_{mna} =$ number of actions, $\epsilon_0 = 0.001$, $\alpha = 0.1$, $\nu = 5$, $\theta_{GA} = 50$, $\chi = 0.8$, $\mu = 0.04$, $\theta_{del} = 50$, $\delta = 0.1$, $doGASubsumption = yes$, $doActionSetSubsumption = no$, $\theta_{sub} = 50$, $r_0 = 0.1$, $s_0 = 0.6$, $Reward = 1000/0$. These are fairly standard settings for real world classification problems, so we have not tuned the parameters specifically for any particular problem. The parameters belonging to the reduction algorithms are set as follows: $\theta_{reduct} = 10$, $\epsilon_{reduct} = 0.1$, $p_{reduct} = 999$.

Table 1, first column, shows the high number of rules evolved by XCS after training. In most of the datasets, this number of rules is even higher than the number of instances of the

Table 1: Number of rules obtained after training XCS (first column) and after applying the reduction algorithms W, D1 and D2.

Dataset	XCS	W	D1	D2
bpa	3253,6	89,6	93,4	1134,1
bps	6061,1	241,6	682,7	1060,7
bre	3885,8	63,9	68,6	1476,7
gls	4142,6	61,3	69,4	848,4
irs	1141,0	12,9	10,0	480,9
mmg	5900,7	67,8	225,4	2109,5
pmi	4234,2	157,6	196,9	1110,0
tao	788,2	46,9	54,8	209,9
veh	5776,8	169,5	282,7	705,6
wne	5206,9	22,9	36,1	1802,0
Avg	4039,1	93,4	172,0	1093,8
%Reduction		97%	96%	71%

problem itself. This does not mean that XCS is not generalizing. In fact, XCS evolves general rules but most of them are redundant and overlap partially with others. This is a consequence of the hyperrectangle representation coupled with the crossover and mutation operators. The reduction algorithms are all able to reduce significantly this high number of rules. Specially, the W and D1 algorithms are the most effective algorithms in terms of reduction. They reduce the population in a percentage of 97% and 96% respectively. For example, in the iris problem, XCS obtains 1141 rules in average, while the reduction algorithms W and D1 obtain 12,9 and 10 rules respectively. This is a reasonable number of rules when considering the explanatory ability of the system. Moreover, the type of explanation obtained by these reduced sets of rules is similar to that obtained by decision trees like C4.5 (not shown for brevity), providing a more meaningful explanation than the ruleset without reduction. The reduction of D2 is less pronounced. In average, it performs a reduction of 71%, which although being significant, is not sufficient if we use the final ruleset for interpretation of human experts. For example, in the iris dataset, we obtain 480,9 rules, which is still too high to have an easy explanation of the knowledge.

After seeing the benefits of the reduction algorithms for the explanatory ability of the system, let's analyze if they cause a significant degradation in classification accuracy. For this purpose, we have applied the compact rulesets to the test sets of the ten-fold cross-validation experiment and we have compared the classification accuracy to that obtained by raw XCS. The results are shown in table 2. Observe that in all cases, the reduction algorithms obtain a loss in classification accuracy. And in most of these cases, the degradation is statistically significant according to a two-tailed paired t-test at 99% confidence level. This loss is less pronounced in the W algorithm, with a mean classification accuracy of 74,60% compared to the 80,19% of raw XCS. The D2 algorithm is worse than W, but better than D1 because it performs less reduction. In fact, the degradation in classification accuracy is reasonable since XCS has been trained to maximize accuracy with a high number of rules, so that reducing it a posteriori causes degradation.

We also compare the CPU time of each of these algorithms. Table 3 shows the cost of the reduction algorithms related to the algorithm that spends the minimum CPU time, which always corresponds to the D2 algorithm. Note that D1 and D2 are almost equivalent in this aspect, while the W algorithm is significantly the slowest algorithm. For example, in the bps dataset it takes 356 minutes to reduce the ruleset (computed with a Pentium IV at 1.5 GHz).

Table 2: Classification accuracy (%) after training XCS and after applying the reduction algorithms W, D1 and D2. Statistical differences are shown according to a two-tailed t-test at 99% confidence level. A • is shown when the classification accuracy of the reduction algorithm is statistically different from that of XCS.

Dataset	XCS	W	D1	D2
bpa	61,80	60,04	52,63 •	52,68
bps	80,74	71,79 •	71,72 •	71,71 •
bre	95,70	92,70 •	92,99 •	94,40
gls	71,22	62,87 •	56,52 •	59,27 •
irs	95,33	94,00	92,67	92,67
mmg	61,51	49,61 •	53,72 •	58,74
pmi	72,43	68,50	63,71 •	67,33 •
tao	92,95	91,47 •	86,17 •	86,49 •
veh	74,54	67,80 •	58,56 •	59,50 •
wne	95,67	87,19 •	90,13	85,45 •
Avg	80,19	74,60	71,88	72,82

Table 3: Cost of Reduction Algorithms. The cost of the W and D1 algorithms are related to D2. The time of D2 is given in minutes.

Dataset	W (W/D2)	D1 (D1/D2)	D2 (min)
bpa	1739,0	1,05	0,0143
bps	5223,8	1,14	0,0682
bre	624,7	1,03	0,0270
gls	1891,7	1,03	0,0167
irs	143,9	1,10	0,0028
mmg	98,14	1,02	0,0540
pmi	3876,39	1,05	0,0250
tao	551,13	1,40	0,0022
veh	6365,99	1,08	0,0422
wne	341,06	1,03	0,0332
Avg	2847,98	1,07	0,0285

The way in which the W algorithm computes the reduction makes this algorithm to spend such CPU time. Note that the W algorithm needs the computation of the classification accuracy of several sets of rules. In comparison, the D1 and D2 algorithms only need to compute this once. In just one step, the algorithm marks the useful classifiers. Therefore, considering also CPU time, the D1 algorithm offers a reasonable alternative to the W algorithm.

4.1 The effect of higher training time

The results shown in the previous section highlight a significant degradation of classification accuracy after reduction. To avoid this degradation, we have performed an experiment with the Wisconsin breast cancer dataset, increasing the number of training iterations to 2,000,000 and then applying again the reduction algorithms. The results are shown in table 4.

Comparing these results with those in tables 2 and 1, we observe that the classification accuracy of raw XCS is very similar in both cases but the number of rules is less with 2,000,000 iterations. Thus, increasing the number of training iterations does not influence the classification accuracy. But although the classification accuracy does not increase, the population achieved by XCS with higher training iterations is different; with higher iterations XCS has

Table 4: Results of XCS (classification accuracy and number of rules) training XCS for 2,000,000 explore iterations in the Wisconsin breast cancer dataset.

	Acc (%)	# rules
XCS	95,42	1875,0
W	93,71	26,0
D1	94,12	36,2
D2	92,53	696,0

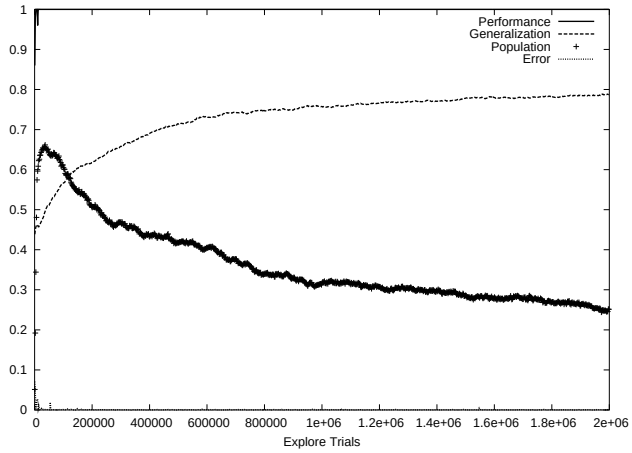

Figure 1: Learning curve of XCS along the explore iterations. The curves show respectively the performance (classification accuracy), the generalization of the rules, the number of rules and the system's error.

more time to stabilize the population and obtain a more generalized set of rules. That is why the number of rules after training for 2,000,000 iterations is less than with 100,000 iterations. This behavior is shown in figure 1. As a consequence, the number of rules after applying the reduction algorithms is also less with 2,000,000 training iterations than with the previous results. The nice effect that arises here is that the classification accuracy after applying the reduction algorithms is not degraded as much as in the previous results (these results also agree with those in [13]). A two-tailed paired t-test at 99% confidence level revealed no significant difference between the classification accuracy of XCS and that of each of the reduction algorithms. Therefore, increasing the number of training iterations does not mean higher classification accuracy of XCS, but implies better reliability of the reduction algorithms. These results also apply to the rest of the datasets, which are not shown for brevity. The only drawback of such high number of iterations is the cost of the algorithm. It takes 142 minutes to train XCS for 2,000,000 iterations on a single fold of the Wisconsin breast cancer dataset (running on a Pentium IV at 1.5 GHz).

5 Conclusions

This paper arises the need of reducing the ruleset evolved by XCS classifier system in order to achieve a good explanatory ability. This is specifically important when XCS is applied to real-world datasets, where the hyperrectangle representation is used. The paper compares different approaches of reduction algorithms in several datasets. Wilson's algorithm shows a high degree of reduction and the fewest degradation of classification accuracy. Its drawback is that it consumes higher CPU time than the D1 and D2 algorithms. The D1 algorithm offers a reasonable reduction with better CPU time, although it gets less classification accuracy

than Wilson's algorithm. Training XCS for a high number of iterations does not improve the classification accuracy but stabilizes the ruleset so that the reduction algorithms have better reliability. In this case, there is not a significant loss of classification accuracy, and the differences between W and D1 are less pronounced (in terms of classification accuracy). A future consideration of this work is the use of reduction algorithms during training, so that we can keep a compact ruleset during learning, which could consequently reduce the training time.

Acknowledgments

The authors agree the support of *Enginyeria i Arquitectura La Salle*, as well as the support of *Ministerio de Ciencia y Tecnología* under project TIC2002-04036-C05-03.

References

[1] Ester Bernadó Mansilla and Josep M. Garrell Guiu. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

[2] Ester Bernadó Mansilla, Xavier Llorà Fàbrega, and Josep M. Garrell Guiu. XCS and GALE: a Comparative Study of Two Learning Classifier Systems on Data Mining. In *Advances in Learning Classifier Systems, 4th International Workshop*, volume 2321 of *Lecture Notes in Artificial Intelligence*, pages 115–132. Springer, 2002.

[3] C.L. Blake and C.J. Merz. UCI Repository of machine learning databases, [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. University of California, Irvine, Department of Information and Computer Sciences, 1998.

[4] M.V. Butz and S.W. Wilson. An algorithmic description of XCS. In P.L. Lanzi, W. Stolzmann, and S.W. Wilson, editors, *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, volume 1996 of *Lecture Notes in Artificial Intelligence*, pages 253–272. Springer, 2001.

[5] Thomas G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

[6] Phillip William Dixon, David W. Corne, and Martin John Oates. A Ruleset Reduction Algorithm for the XCS Learning Classifier System. In *Fifth International Workshop on Learning Classifier Systems (IWLCS'2002)*, 2002.

[7] Tim Kovacs. XCS Classifier System Reliably Evolves Accurate, Complete and Minimal Representations for Boolean Functions. In *Soft Computing in Engineering Design and Manufacturing*, pages 59–68. Springer-Verlag, 1997.

[8] Tim Kovacs. Deletion Schemes for Classifier Systems. In *Proceedings of the Genetic and Evolutionary Computation Conference, (GECCO-99)*, pages 329–336. Morgan Kaufmann, 1999.

[9] Christopher Stone and Larry Bull. For Real! XCS with Continuous-Valued Inputs. *Evolutionary Computation*, 11(3):299–336, 2003.

[10] S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[11] Stewart W. Wilson. Generalization in the XCS Classifier System. In *Genetic Programming: Proceedings of the Third Annual Conference*, pages 665–674. Morgan Kaufmann, 1998.

[12] Stewart W. Wilson. Get Real! XCS with Continuous-Valued Inputs. In L. Booker, S. Forrest, M. Mitchell, and R Riolo, editors, *Festschrift in Honor of John H. Holland*, pages 111–121. Center for the Study of Complex Systems, University of Michigan, 1999.

[13] Stewart W. Wilson. Compact Rulesets from XCSI. In *Advances in Learning Classifier Systems, 4th International Workshop*, volume 2321 of *Lecture Notes in Artificial Intelligence*, pages 197–210. Springer, 2002.