

Investigating the Impact of Independent Rule Fitnesses in a Learning Classifier System

Michael Heider^(✉), Helena Stegherr, Jonathan Wurth, Roman Sraj,
and Jörg Hähner

Universität Augsburg, Am Technologiezentrum 8, Augsburg, Germany
{michael.heider,helena.stegherr,jonathan.wurth,
roman.sraj,jorg.hahner}@uni-a.de

Abstract. Achieving at least some level of explainability requires complex analyses for many machine learning systems, such as common black-box models. We recently proposed a new rule-based learning system, SupRB, to construct compact, interpretable and transparent models by utilizing separate optimizers for the model selection tasks concerning rule discovery and rule set composition. This allows users to specifically tailor their model structure to fulfil use-case specific explainability requirements. From an optimization perspective, this allows us to define clearer goals and we find that—in contrast to many state of the art systems—this allows us to keep rule fitnesses independent. In this paper we investigate this system’s performance thoroughly on a set of regression problems and compare it against XCSF, a prominent rule-based learning system. We find the overall results of SupRB’s evaluation comparable to XCSF’s while allowing easier control of model structure and showing a substantially smaller sensitivity to random seeds and data splits. This increased control can aid in subsequently providing explanations for both training and final structure of the model.

Keywords: Rule-based learning · Learning classifier systems · Evolutionary machine learning · Interpretable models · Explainable AI

1 Introduction

The applicability of decision making agents utilizing machine learning methods in real-world scenarios depends not only on the accuracy of the models, but equally on the degree to which explanations of the decisions can be provided to the human stakeholders. For example, in an industrial setting, experienced machine operators often rather rely on their own knowledge instead of on—in their eyes—unsubstantiated recommendations of the model going against that knowledge. This problem is exacerbated as it is inevitable that the model is not perfect in every detail, especially when the learning task is complex and the available training data limited.

To still make use of the advantages of recommendations made by digital agents, increasing the trust of stakeholders in the predictions is essential. It includes providing explanations of the processes involved to produce these, as well as of the entire model. This can get to a point where easily explainable models are preferred over better performance with higher complexity. Rule-based learners such as Learning Classifier Systems (LCSs) are well suited in these settings as they facilitate extensive explanations [10].

LCSs [25] are inherently transparent and interpretable rule-based learners that make use of a finite set of if-then rules to compose their models. Each rule contains a simpler, more comprehensible submodel, related to specific areas of the feature space. The conditions under which rules apply are optimized during the training process, commonly by an evolutionary algorithm. There are two main styles of LCSs: Pittsburgh-style systems, which evolve a population of sets of rules with combined fitnesses (one per set), and Michigan-style systems, which adapt a single set of rules over time with individual fitnesses (one per rule). Therefore, optimization by the evolutionary algorithm is performed differently in the two styles, but always aimed at finding an “accurate and maximally general” [23] set of rules. Explainability requisites are commonly not directly included as optimization targets for the much more frequent Michigan-style systems, though it is to some extent represented under the concept of generality. In Pittsburgh-style systems, the evolutionary algorithm does typically include error and rule set size as targets but it has to optimize the positioning and also the selection of rules. Therefore, each iteration is comprised of several changes to rules in the set which leads to common situations where beneficial changes to a rule are not reflected in a corresponding change to the fitness of the set and might therefore be discarded for the next generation. While the suboptimal positioning of rules might not even decrease the system’s performance, it is, however, a problem when explanations concerning the rule conditions or the training process should be given. Michigan-style systems, on the other hand, often generate and keep a large set of both good and suboptimal rules, in total, far more than required for the given problem. Therefore, they need additional procedures after training, especially *compaction* techniques, to reduce the population to the most important rules and therefore to enhance explainability [16,20].

The first description of a new LCS algorithm, in which the optimization of rule conditions is separated from the composition of rules to form a problem solution, was provided in [11]. This way, rule fitnesses are kept independent from other influences than their direct changes, increasing the *locality*. It also improves the explainability of these quality parameters. Additionally, explainability is improved through the direct control over population sizes and whether good rules should be optimized to be more specific or more general. In this paper, we extend the initial examinations of SupRB, as described in Sect. 3, by evaluating against a modern version of XCSF [19,27], one of the most developed and advanced LCSs, on a variety of different regression datasets (cf. Sect. 4). We find that, as intended, SupRB performs competitively based on hypothesis testing on error distributions as well as Bayesian comparison [4] across datasets, while producing more compact models directly.

2 Related Work

The XCS Classifier System (XCS) is a prominent representative of LCSs. Its many derivatives and extensions are capable of solving all three major learning tasks [25]. In the context of this paper, the most notable extensions are those concerned with applicability to real-valued problem domains and supervised function approximation. In terms of real-valued problem domains, this means replacing binary matching function with interval-based ones [26]. For supervised function approximation, XCSF was designed [27]. It replaces the constant predicted payoff with a linear function. To further enhance the performance, more complex variants were introduced to replace linear models and interval-based matching functions [6, 13], however, at the cost of overall model transparency.

LCSs are commonly considered as transparent or interpretable by design, as are other rule-based learning systems, and naturally relate to human behaviour. In contrast, other systems require extensive post-hoc methods, such as visualisation or model transformation, to reach explainability. Even though LCS can be seen as inherently transparent, there can be factors that reduce these capabilities. They may arise through the encodings used, the number of rules in general and the complexity introduced by using complex matching functions or submodels in the individual rules [3].

Controlling these limitations in LCSs is typically done by design but can incorporate designated post-hoc methods. Post-hoc methods, especially visualisation techniques for classifiers, can improve the interpretability of the model [15, 17, 24]. However, they have to be devised or adapted to the specific needs of the problem at hand and the model itself, which requires time and expertise. Controlling transparency by design can therefore be beneficial in some cases. While some factors, for example problem-dependent complex variables/features, restrict interpretability and can hardly be influenced, other factors can compensate for these issues. This means the design must consider understandable matching functions and predictive submodels, without foregoing an adequate predictive power.

Another aspect strongly related to the interpretability of LCS models is the size of the resulting rule sets, e.g. smaller sets facilitate direct visual inspection and require less subsequent analysis. Controlling this size is handled differently in Pittsburgh-style and Michigan-style systems. Pittsburgh-style LCSs utilize the fitness function of the optimization algorithm, which often incorporates different objectives, i.e. accuracy and number of rules. A prominent example is GAssist [2], where accuracy and minimum description length form a combined objective and an additional penalty is given if the rule set size gets too small. Michigan-style systems, on the other hand, do not control the rule set size by means of the fitness function, as large populations are often beneficial for the training process. During the training, subsumption can be performed to merge two rules where one fully encompasses the other. Compaction is a post-hoc method to reduce the size of the rule set after training by removing redundant rules without decreasing the prediction accuracy [16, 28]. However, most compaction methods are purely designed for classification.

3 The Supervised Rule-Based Learning System

We recently proposed [11] a new type of LCS with interchanging phases of rule discovery and solution composition, the Supervised Rule-based Learning System (SupRB). The first phase optimizes rule conditions independently of other rules, discovering a diverse pool of well proportioned rules. Subsequently, in the second phase, another optimization process selects a subset of all available rules to compose a good (accurate yet small) solution to the learning task. In contrast to other LCSs, we thus separate the *model selection* objectives of finding multiple well positioned rules (with a tradeoff between local prediction error and matched volume) and selecting a set of these rules for our final model. That allows us to predict arbitrary inputs with minimal error while the set of rules is as small as possible to keep transparency and interpretability high. As it can be difficult to determine how many rules would need to be generated before a good solution can be composed from them, the two phases are alternated until some termination criterion, e.g. a certain number of iterations, is reached (cf. Algorithm 1). Note that, in contrast to Pittsburgh-style systems, rules added to the pool remain unchanged and will not be removed throughout the training process. An advantage of alternating phases is the ability to steer subsequent rule discoveries towards exploring regions where no or ill-placed rules are found, based on information from the solution composition phase.

Algorithm 1. SupRB’s main loop

```

1: pool  $\leftarrow \emptyset$ 
2: elitist  $\leftarrow \emptyset$ 
3: for  $i \leftarrow 1, n\_iter$  do
4:   pool  $\leftarrow$  pool  $\cup$  DISCOVER RULES(elitist)
5:   elitist  $\leftarrow$  COMPOSE SOLUTION(pool, elitist)
6: end for
7: return elitist

```

Insights into decisions are a central aspect of SupRB, therefore, its model is kept as simple and interpretable as possible [11]:

1. Rules’ conditions use an interval based matching: A rule k applies for example x iff $x_i \in [l_{k,i}, u_{k,i}] \forall i$ with l being the lower and u the upper bounds.
2. Rules’ submodels $f_k(x)$ are linear. They are fit using linear least squares with a l2-norm regularization (Ridge Regression) on the subsample matched by the respective rule.
3. When mixing multiple rules to make a prediction, a rule’s experience (the number of examples matched during training and therefore included in fitting the submodel) and in-sample error are used in a weighted sum.

In general, a large variety of methods can be used to discover new rules, but for this paper, we utilize an evolution strategy (ES). The overall process

Algorithm 2. SupRB’s Rule Discovery

```

1: procedure DISCOVER RULES(elitist)
2:   rules  $\leftarrow \emptyset$ 
3:   for  $i \leftarrow 1, n\_rules$  do  $\triangleright (1, \lambda)$ -ES for each new rule
4:     candidate, proponent  $\leftarrow$  INIT RULE(elitist)
5:     repeat
6:       children  $\leftarrow \emptyset$ 
7:       for  $k \leftarrow 1, \lambda$  do
8:         children  $\leftarrow$  children  $\cup$  MUTATE(proponent)
9:       end for
10:      proponent  $\leftarrow$  child with highest fitness
11:      if candidate’s fitness  $<$  proponent’s fitness then
12:        candidate  $\leftarrow$  proponent
13:         $j \leftarrow 0$ 
14:      else
15:         $j \leftarrow j + 1$ 
16:      end if
17:    until  $j = \delta$ 
18:    rules  $\leftarrow$  rules  $\cup$  candidate
19:  end for
20:  return rules
21: end procedure

```

is displayed in Algorithm 2. While during a rule discovery phase typically multiple rules are discovered and added, this happens independently (and can be parallelized) in multiple $(1, \lambda)$ -ES runs. The initial candidate and parent rule is placed around a roulette-wheel selected training example, assigning higher probabilities to examples whose prediction showed a high in-sample error in the current (intermediate) solution (or *elitist*). The non-adaptive mutation operator samples a halfnormal distribution twice per dimension to move the parent’s upper and lower bounds further from the center by the respective values. This is repeated to create λ children. From these, the fittest individual is selected based on its in-sample error and the matched feature space volume as the new parent. If it displays a higher fitness than the candidate it becomes the new candidate. Specifically, the fitness is calculated as

$$F(o_1, o_2) = \frac{(1 + \alpha^2) \cdot o_1 \cdot o_2}{\alpha^2 \cdot o_1 + o_2}, \quad (1)$$

with

$$o_1 = \text{PACC} = \exp(-\text{MSE} \cdot \beta), \quad (2)$$

and

$$o_2 = V = \prod_i \frac{u_i - l_i}{\min_{x \in \mathcal{X}} x_i - \max_{x \in \mathcal{X}} x_i}. \quad (3)$$

The base form (cf. Eq. (1)) was adapted from [29], where it was combining two objectives in a feature selection context. The Pseudo-Accuracy (PACC), Eq. (2),

squashes the Mean Squared Error (MSE) of a rule’s prediction into a $(0, 1]$ range, while the volume share $V \in [0, 1]$ (cf. Eq. 3) of its bounds is used as a generality measure. The parameter β controls the slope of the PACC and α weighs the importance of o_1 against o_2 . We tested multiple values for β and found $\beta = 2$ to be a suitable default. For α , 0.05 can be used in many problems (hyperparameter tuning for the datasets in this paper selected it in 3 out of 4 cases) but, ultimately, the value should always depend on the model size requirements, which are task dependent. If the candidate has not changed for δ generations, the optimization process is stopped and this specific elitist is added to the pool. This process of discovering a new rule and adding it to the pool of rules is repeated until the set number of rules has been found. We want to stress that this optimizer is not meant to find a single globally optimal rule as in typical optimization problems, but rather find optimally placed rules so that for all inputs a prediction can be made that is more accurate than a trivial model, i.e. simply returning the mean of all data. Therefore, independent evolution is advantageous.

Algorithm 3. SupRB’s Solution Composition

```

1: procedure COMPOSE SOLUTION(pool, elitist)
2:   population  $\leftarrow$  elitist
3:   for  $i \leftarrow 1, \text{pop\_size}$  do
4:     population  $\leftarrow$  population  $\cup$  INIT SOLUTION()
5:   end for
6:   for  $i \leftarrow 1, \text{generations}$  do
7:     elitists  $\leftarrow$  SELECT ELITISTS(population)
8:     parents  $\leftarrow$  TOURNAMENT SELECTION(population)
9:     children  $\leftarrow$  Crossover(parents) ▷ 90% probability n-point
10:    population  $\leftarrow$  MUTATE(children) ▷ probabilistic bitflip
11:    population  $\leftarrow$  population  $\cup$  elitists
12:   end for
13:   return best solution from population
14: end procedure

```

In the solution composition phase, a genetic algorithm (GA) selects a subset of rules from the pool to form a new solution. As with the rule discovery, many optimizers could be used and a few have already been tested in [30], finding that the GA is a suitable choice. Solutions are represented as bit strings, signalling whether a rule from the pool is part of the solution. The GA uses tournament selection to select groups of two solutions and combines two parents by using n -point crossover with a default crossover probability of 90%. Then, mutation is applied to the children, flipping each bit with a probability determined by the mutation rate. The children and some of the fittest parents (*elitism*) form the new population. The number of elitists depends on the population size of the GA, but in our experiments, we found 5 or 6 to work best with a population size of 32. Solution fitness is also based on Eq. (1). Here, the solution’s in-sample mean squared error and its *complexity*, i.e. the number of rules selected, are used

as first and second objective, respectively. Note that each individual in the GA always corresponds to a subset of the pool. Rules that are not part of the pool can not be part of a solution candidate and rules remain unchanged by the GA's operations.

SupRB is conceptualised and designed as a regressor. This is reflected in both the description above and the evaluation in the following section. However, we want to propose how the system could be adapted easily towards solving classification problems: The linear submodels would need to be replaced with an appropriate classifier, either simply a constant model, logistic regression or a more complex model if the explainability requirements allowed that. Additionally, the fitness functions would need to use accuracy (or an appropriate scoring for imbalanced data) instead of PACC and MSE.

4 Evaluation

For our evaluation of the proposed system, we compare SupRB to a recent XCSF¹ [19,27] with hyperrectangular conditions and linear submodels (with recursive least squares updates [14]), as they closely correspond to the conditions and submodels used in SupRB. We acknowledge that some better performing conditions, e.g. hyperellipsoids [7], have been proposed for XCSF, however, we consider them less interpretable in high dimensional space for the average user.

4.1 Experiment Design

SupRB is implemented² in Python 3.9, adhering to *scikit-learn* [18] conventions. Input features are transformed into the range $[-1, 1]$, while the target is standardized. Both transformations are reversible but improve SupRB's training process as they help preventing rules to be placed in regions where no sample could be matched and remove the need to tune error coefficients in fitness calculations, respectively. Based on our assumptions about the number of rules needed, 32 cycles of alternating rule discovery and solution composition are performed, generating four rules in each cycle for a total of 128 rules. For the ES we selected a λ of 20. Additionally, the GA is configured to perform 32 iterations with a population size of 32. To tune some of the more sensitive parameters, we performed a hyperparameter search using a Tree-structured Parzen Estimator in the Optuna framework [1] that optimizes average solution fitness on 4-fold cross validation. We tuned datasets independently for 256 iterations per tuning process. For XCSF we followed the same process, selecting typical default values³ [19] and tuning the remaining parameters independently on the four datasets using the same setup as before. The final evaluation, for which we report results in Sect. 4.2, uses 8-split Monte Carlo cross-validation, each with 25 % of samples

¹ <https://github.com/rpreen/xcsf>, <https://doi.org/10.5281/zenodo.5806708>.

² <https://github.com/heidmic/suprb>, <https://doi.org/10.5281/zenodo.6460701>.

³ <https://github.com/rpreen/xcsf/wiki/Python-Library-Usage>.

reserved as a validation set. Each learning algorithm is evaluated with 8 different random seeds for each 8-split cross-validation, resulting in a total of 64 runs.

We evaluate on four datasets part of the UCI Machine Learning Repository [9]. The Combined Cycle Power Plant (CCPP) [12,22] dataset shows an almost linear relation between features and targets and can be acceptably accurately predicted using a single rule. Airfoil Self-Noise (ASN) [5] and Concrete Strength (CS) [31] are both highly non-linear and will likely need more rules to predict the target sufficiently. The CS dataset has more input features than ASN but is easier to predict overall. Energy Efficiency Cooling (EEC) [21] is another rather linear dataset, but has a much higher input features to samples ratio compared to CCPP. It should similarly be possible to model it using only few rules.

4.2 Results

In our experiments we find that XCSF and SupRB achieve comparable results. Table 1 presents the dataset-specific performance in detail. All entries are calculated on 64 runs per dataset (cf. Sect. 4.1). As both systems were trained for standardized targets, we denote the results for the mean (across runs) mean squared errors (MSE) and their standard deviation (STD) as MSE_σ and STD_σ , respectively. Standardized targets allow better comparison between the datasets as results are on a more similar scale. Additionally, as many real world datasets are normally distributed, this should lighten the need to carefully hand tune the balance between solution complexity and error. Note that predictions of both models can always be retransformed into the original domain. Subsequently, MSE_{orig} references the mean MSE in units of the original dataset-specific target domain. Although this column is less helpful for cross dataset performance interpretations, it allows comparison to other works on the same data. We found that, on two datasets (CCPP and ASN), XCSF shows a better performance, albeit only slightly for CCPP, that can be confirmed through hypothesis testing (Wilcoxon signed-rank test using a confidence level of 5%). Contrastingly, for the CS dataset, the hypothesis could not be rejected. Thus, although SupRB shows a slightly lower mean MSE, this is not statistically significant. For the EEC dataset SupRB outperforms XCSF.

We found that SupRB’s runs had a similar (to each other) performance much more consistently than XCSF’s. This is shown by STD_σ (cf. Table 1) and specifically illustrated in Fig. 1, which shows the distribution of test errors across all 64 runs. For three of the four datasets, XCSF shows some strong outliers that go against its remaining performances. Additionally, the majority of runs is also further distributed around the mean and median values. We assume that this is largely due to the stochastic iterative nature of training in XCSF. For the CCPP dataset (Fig. 1a) no outliers were produced by XCSF and overall performance is quite similar across runs. This is especially noticeable when comparing the distribution to those on the other datasets. In fact, the runs are so similar (even across models) that it is hard to make any analysis on this scale. Although, XCSF slightly outperformed SupRB on average on CCPP, as confirmed by statistical

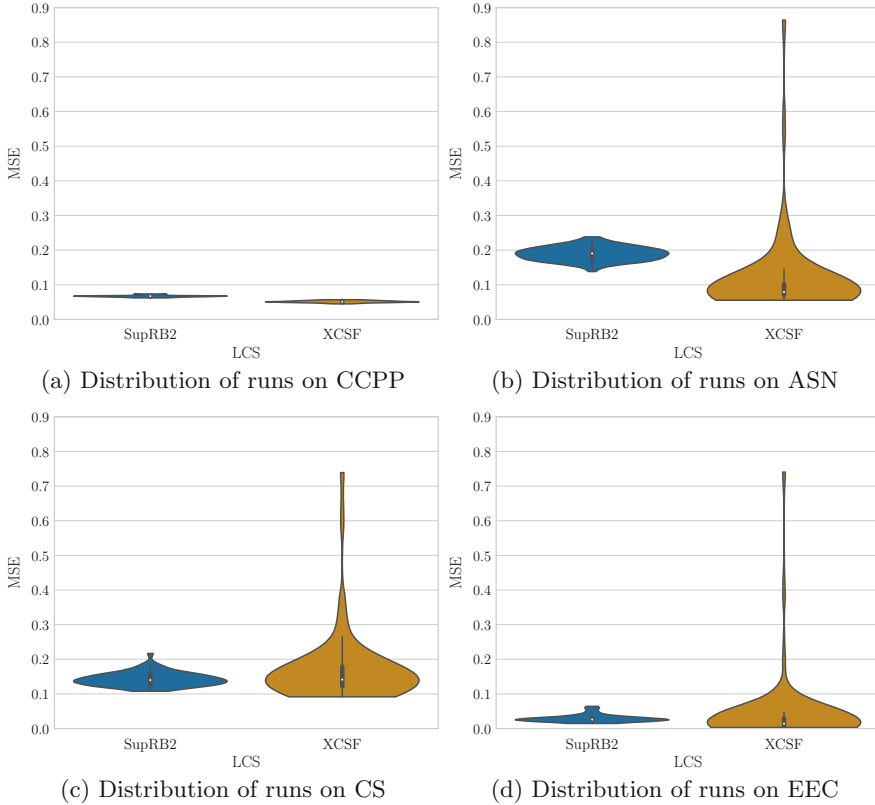


Fig. 1. Distribution of runs' errors on an equal scale.

testing, we can assume that this advantage is likely not practically significant. From a graphical perspective (cf. Fig. 1c), SupRB seems to produce more desirable models on CS, even if the hypothesis testing remained ambiguous. On EEC, XCSF achieves a slightly better median MSE performance ($\text{Median}_{\text{XCSF}}: 0.014$; $\text{Median}_{\text{SupRB}}: 0.026$), however, its mean MSE is poorer due to badly performing runs. Regardless, the overall performance can be viewed as rather close, although both sets of runs are clearly not following the same distribution. As SupRB's and XCSF's models were trained on the same random seeds and cross-validation splits, we can conclude that SupRB is overall more reliable even if not necessarily better.

For SupRB we directly control the size (number of rules; *complexity*) of the global solution via the corresponding fitness function used in the GA. Table 2 shows the complexities of the 64 runs per dataset. Note that the highest theoretical complexity is 128, as we did only add 128 rules to the pool. We find that, although theoretically a single rule is able to predict CCPP well, the optimizer prefers to use at least two but at most four rules, achieving slightly better errors

Table 1. Overview of the experimental test data results of 64 runs per dataset rounded to four decimals. MSE_{orig} and MSE_{σ} give the means of the mean squared errors (MSE) in the dataset’s original or a standardised target space, respectively. Similarly, STD_{σ} displays the standard deviation of MSEs in standardised space. Highlighted in bold are the models where a 5% significance Wilcoxon signed-rank test rejected the null hypothesis of equivalent distributions and the mean was better.

	CCPP			ASN		
	MSE_{orig}	MSE_{σ}	STD_{σ}	MSE_{orig}	MSE_{σ}	STD_{σ}
XCSF	0.8745	0.0512	0.0028	0.7930	0.1150	0.1195
SupRB	1.1433	0.0669	0.0027	1.3079	0.1896	0.0199
	CS			EEC		
	MSE_{orig}	MSE_{σ}	STD_{σ}	MSE_{orig}	MSE_{σ}	STD_{σ}
XCSF	2.8291	0.1694	0.1043	0.3660	0.0385	0.1032
SupRB	2.3779	0.1424	0.0199	0.2776	0.0292	0.0107

Table 2. Overview of the solution complexities (number of rules in the solution proposed by SupRB or the final macro-classifier count in an XCSF population, respectively) across 64 runs per dataset.

	SupRB				XCSF			
	CCPP	ASN	CS	EEC	CCPP	ASN	CS	EEC
Mean	2.65	26.42	22.31	12.81	2253.28	962.03	562.81	1028.78
St. dev	0.62	2.47	2.60	1.71	24.70	9.17	11.73	14.90
Median	3	27	22	13	2250	962	562	1026
Min	2	19	17	9	2202	934	530	994
Max	4	30	30	17	2301	980	593	1068

than with a singular linear model. As expected, the solutions to the two highly non-linear datasets (ASN and CS) do feature considerably more rules. EEC again was solved with fewer rules, speaking to its more linear nature, although with more than CCPP, for which a linear solution exists. Standard deviations of complexities increase as the mean increases and the median stays close to the mean.

XCSF seems to have fallen into a cover-delete-cycle where rules did not stay part of the population for long. Covering is a rule generation mechanism that creates a new rule whenever there were too few matching rules. The deletion mechanism removes rules when the population is too full, as there exists a hyperparameter-imposed maximum population size. In our tuning, we did tune both the number of training steps and the maximum population size (among the many other parameters of XCSF) and find that post-training populations are at or around the maximum population size. XCSF’s hyperparameter tuning

opted for much larger populations than the typical rule of thumb of using ten times as many rules as would be expected (from domain knowledge or prior modelling experience) for a good problem solution [23]. Additionally, upon deeper inspection, we found that the rules were typically introduced late in the training process, however, the system error did not change in a meaningful manner long before that point. Note that we did utilize subsumption in the EA. This mechanism prevents the addition of a newly produced rule to the population when it is fully engulfed by a parent rule and instead increases the parents numerosity parameter. A rule with numerosity n counts as n rules with numerosity 1 towards the maximum population size limit. Subsumption thus theoretically decreases the actual number of classifiers in our population. However, in our experiments the cover-delete-cycle seems to have rendered this mechanism useless.

It is reasonably possible that SupRB’s performance would improve in some cases if the pressure to evolve smaller rule sets was lower. However, as explainability suffers with large rule sets, we think that the presented solutions strike an acceptable balance. Afterall, XCSF’s solutions were substantially larger even after applying a simple compaction technique of removing rules with an experience of 0 from the final population. This compaction method removed on average about 10% of rules from the run’s populations. Table 2 reports the complexity results after compaction. However, we acknowledge that a variety of compaction techniques exists for classification problems [16] that could in some cases potentially be adjusted for the use within regression tasks. Likely, SupRB and XCSF find themselves at different points on the Pareto front between error and complexity. However, in SupRB we do not need to rely on additional post-processing but can solve this optimization problem directly and, importantly, balance the tradeoff of prediction error and rule set complexity against user needs, whereas compaction mechanisms are typically designed to decrease complexity only in a way as to not increase the LCS’s system error [16].

Beyond dataset-specific performances, we would like to find a more general answer to the question whether the newly proposed SupRB does perform similarly to the well established XCSF. This would indicate that we can find a good LCS model even without the niching mechanisms employed by XCSF’s rule fitness assignment. To find an initial answer based on the performed experiments we use a Bayesian model comparison approach [4] using a hierarchical model [8] that jointly analyses the cross-validation results across multiple random seeds and all four datasets. We assume a region of practical equivalence of $0.01 \cdot \sigma_{\text{dataset}}$.

$$p(\text{SupRB} \ll \text{XCSF}) \approx 63.4\%$$

$$p(\text{SupRB} \equiv \text{XCSF}) \approx 8.5\%$$

$$p(\text{SupRB} \gg \text{XCSF}) \approx 28.1\%$$

where:

- $p(\text{SupRB} \ll \text{XCSF})$ denotes the probability that SupRB performs worse (achieving a higher MSE on test data),
- $p(\text{SupRB} \equiv \text{XCSF})$ denotes the probability that both systems achieve practically equivalent results and

- $p(\text{SupRB} \gg \text{XCSF})$ denotes the probability that SupRB performs better (achieving a lower MSE on test data).

From these results we clearly can not make definitive assessments that XCSF is stronger than SupRB. While it might outperform SupRB in less than two thirds of cases, it also will be outperformed in almost a third of cases. [4] suggest thresholds of 0.95, 0.9 or 0.8 for probabilities to make automated decisions. The specific value needs to be chosen according to the given context. We did not perform the same analysis for the rule set sizes as the results are quite clear with SupRB being the system very likely producing much smaller rule sets. Overall, we can conclude that no clear decision can be made and that the newly developed (and to be improved in the future) SupRB should be considered an equal to the well established XCSF.

Table 3. Exemplary rule generated by SupRB on CS dataset. The target is the concrete compressive strength. The original space intervals denote the area matched by the rule in terms of the original variable scales, while the intervals in feature spaces are scaled into $[-1, 1]$ and help perceiving rule generality at a glance. Coefficients denote the weight vector used for the linear model.

Input variable	Original Space		Feature Space σ		
	Interval		Interval	Coefficient	
Cement [kg/m ³]	[104.72, 516.78]		[-0.99, 0.89]	2.38	
Blast Furnace Slag [kg/m ³]	[0, 359.40]		[-1.00, 1.00]	2.29	
Fly Ash [kg/m ³]	[13.45, 200]		[-0.87, 1.00]	0.68	
Water [kg/m ³]	[122.64, 244.80]		[-0.99, 0.96]	-1.26	
Superplasticizer [kg/m ³]	[6.02, 24.80]		[-0.63, 0.54]	-0.67	
Coarse Aggregate [kg/m ³]	[950.16, 1145]		[-0.13, 1.00]	0.71	
Fine Aggregate [kg/m ³]	[756.14, 992.60]		[-0.19, 1.00]	0.60	
Age [days]	[18.36, 365]		[-0.90, 1.00]	2.07	
intercept $_{\sigma}$ = 3.9160					
In-sample MSE _{orig}	1.5310	In-sample MSE $_{\sigma}$	0.0917	Experience	84

Table 3 presents a rule trained for the CS dataset. It has an experience (number of matched examples during training) of 84 and matched another 31 examples during testing. It is part of a model consisting of 23 rules with experiences of 7 to 240 with a mean experience of 54.17 ± 55.63 . The rules were, thus, either rather general or rather specific with this rule being on the more general side. Upon closer inspection, for 5 of the 8 dimensions of CS the rule matches most of the available inputs (being maximally general on the “Blast Furnace Slag” input variable). For the transformed input space (feature space) that is scaled to an interval of $[-1, 1]$ this can easily be seen without any knowledge about the datasets structure, although it is likely that users of the model will have enough domain knowledge to be able to derive this directly from the intervals in the original space. It can also be assumed that these users will generally prefer to inspect the rule in that representation. High concentrations of “Water” and “Superplasticizer” have negative effects on the compressive strength of the

concrete for the aforementioned value ranges, while higher concentrations of “Cement”, “Blash Furnace Slag” and “Age” of the mixture positively influence its compressive strength. The other three input variables have positive but less pronounced effects. Overall, rule inspection offers some critical insights into the decision making process and can be done fairly easily based on the rule design and the low number of rules per solution.

5 Conclusion

In this paper, we expanded the view on the Supervised Rule-based Learning System (SupRB) with an optimization perspective. We highlighted the advantages of individual rule fitnesses compared to the fitness-sharing approaches typical for other Learning Classifier Systems (LCSs) and discussed our approach to perform LCS model selection using two separated optimizers from that perspective.

To evaluate the system we compared it to XCSF, a well known LCS with a long research history, on four real world regression datasets with different dimensionalities and problem complexities. As one of the greatest advantages of LCS compared to other learning systems is their inherent interpretability and transparency, we limited our study to the use of hyperrectangular conditions and linear models for both systems. After hyperparameter searches for the more sensitive parameters (256 evaluations with 4-fold cross validation), we performed a total of 64 (8 random seeds and 8-fold cross validation with 25% test data) runs of each system on every dataset. We found that, in general, performance is relatively similar. While XCSF showed a statistically better mean test error on two datasets, it was outperformed on one and no statistically significant decision could be made on the fourth dataset. We performed a Bayesian model comparison approach using a hierarchical model and found that no clearly better model can be determined on errors. Solution sizes of SupRB were better than XCSF’s even when applying some form of compaction. Additionally, SupRB was more consistent in its performance across runs. Thus, we conclude that, for now and with future research pending, both systems produce similarly performing models.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, pp. 2623–2631. Association for Computing Machinery, New York (2019). <https://doi.org/10/gf7mzz>
2. Bacardit, J.: Pittsburgh genetics-based machine learning in the data mining era: representations, generalization, and run-time. Ph.D. thesis, PhD thesis, Ramon Llull University, Barcelona (2004)
3. Barredo Arrieta, A., et al.: Explainable Artificial Intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* **58**, 82–115 (2020). <https://doi.org/10.1016/j.inffus.2019.12.012>

4. Benavoli, A., Corani, G., Demšar, J., Zaffalon, M.: Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. *J. Mach. Learn. Res.* **18**(1), 2653–2688 (2017)
5. Brooks, T., Pope, D., Marcolini, M.: Airfoil self-noise and prediction (1989)
6. Bull, L., O’Hara, T.: Accuracy-based neuro and neuro-fuzzy classifier systems. In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, GECCO 2002*, pp. 905–911. Morgan Kaufmann Publishers Inc., San Francisco (2002)
7. Butz, M.V.: Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system. In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO 2005*, pp. 1835–1842. Association for Computing Machinery, New York (2005). <https://doi.org/10.1145/1068009.1068320>
8. Corani, G., Benavoli, A., Demšar, J., Mangili, F., Zaffalon, M.: Statistical comparison of classifiers through Bayesian hierarchical modelling. *Mach. Learn.* **106**(11), 1817–1837 (2017). <https://doi.org/10.1007/s10994-017-5641-9>
9. Dua, D., Graff, C.: UCI machine learning repository (2017). <https://archive.ics.uci.edu/ml>
10. Heider, M., Nordsieck, R., Hähner, J.: Learning classifier systems for self-explaining socio-technical-systems. In: Stein, A., Tomforde, S., Botev, J., Lewis, P. (eds.) *Proceedings of LIFELIKE 2021 Co-located with 2021 Conference on Artificial Life (ALIFE 2021)* (2021). <https://ceur-ws.org/Vol-3007/>
11. Heider, M., Stegherr, H., Wurth, J., Sraj, R., Hähner, J.: Separating rule discovery and global solution composition in a learning classifier system. In: *Genetic and Evolutionary Computation Conference Companion (GECCO 2022 Companion)* (2022). <https://doi.org/10.1145/3520304.3529014>
12. Kaya, H., Tüfekci, P.: Local and global learning methods for predicting power of a combined gas & steam turbine (2012)
13. Lanzi, P.L., Loiacono, D.: XCSF with neural prediction. In: *2006 IEEE International Conference on Evolutionary Computation*, pp. 2270–2276 (2006). <https://doi.org/10.1109/CEC.2006.1688588>
14. Lanzi, P.L., Loiacono, D., Wilson, S.W., Goldberg, D.E.: Prediction update algorithms for XCSF: RLS, Kalman filter, and gain adaptation. In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO 2006*, pp. 1505–1512. Association for Computing Machinery, New York (2006). <https://doi.org/10.1145/1143997.1144243>
15. Liu, Y., Browne, W.N., Xue, B.: Absumption to complement subsumption in learning classifier systems. In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019*, pp. 410–418. Association for Computing Machinery, New York (2019). <https://doi.org/10.1145/3321707.3321719>
16. Liu, Y., Browne, W.N., Xue, B.: A comparison of learning classifier systems’ rule compaction algorithms for knowledge visualization. *ACM Trans. Evolut. Learn. Optim.* **1**(3), 10:1–10:38 (2021). <https://doi.org/10/gn8gjt>
17. Liu, Y., Browne, W.N., Xue, B.: Visualizations for rule-based machine learning. *Nat. Comput.* (11), 1–22 (2021). <https://doi.org/10.1007/s11047-020-09840-0>
18. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
19. Preen, R.J., Pätzelt, D.: XCSF (2021). <https://doi.org/10.5281/zenodo.5806708>. <https://github.com/rpreen/xcsf>

20. Tan, J., Moore, J., Urbanowicz, R.: Rapid rule compaction strategies for global knowledge discovery in a supervised learning classifier system. In: ECAL 2013: The Twelfth European Conference on Artificial Life, pp. 110–117. MIT Press (2013). <https://doi.org/10.7551/978-0-262-31709-2-ch017>
21. Tsanas, A., Xifara, A.: Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy Build.* **49**, 560–567 (2012). <https://doi.org/10/gg5vzx>
22. Tüfekci, P.: Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *Int. J. Electri. Power Energy Syst.* **60**, 126–140 (2014). <https://doi.org/10/gn9s2h>
23. Urbanowicz, R.J., Browne, W.N.: Applying LCSs. In: *Introduction to Learning Classifier Systems*. SIS, pp. 103–123. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-55007-6_5
24. Urbanowicz, R.J., Granizo-Mackenzie, A., Moore, J.H.: An analysis pipeline with statistical and visualization-guided knowledge discovery for Michigan-style learning classifier systems. *IEEE Comput. Intell. Mag.* **7**(4), 35–45 (2012). <https://doi.org/10.1109/MCI.2012.2215124>
25. Urbanowicz, R.J., Moore, J.H.: Learning classifier systems: a complete introduction, review, and roadmap. *J. Artif. Evolut. Appl.* (2009)
26. Wilson, S.W.: Get Real! XCS with continuous-valued inputs. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) *IWLCS 1999*. LNCS (LNAI), vol. 1813, pp. 209–219. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45027-0_11
27. Wilson, S.W.: Classifiers that approximate functions. *Nat. Comput.* **1**(2/3), 211–234 (2002). <https://doi.org/10.1023/a:1016535925043>
28. Wilson, S.W.: Compact rulesets from XCSI. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) *IWLCS 2001*. LNCS (LNAI), vol. 2321, pp. 197–208. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-48104-4_12
29. Wu, Q., Ma, Z., Fan, J., Xu, G., Shen, Y.: A feature selection method based on hybrid improved binary quantum particle swarm optimization. *IEEE Access* **7**, 80588–80601 (2019). <https://doi.org/10/gnxcfb>
30. Wurth, J., Heider, M., Stegherr, H., Sraj, R., Hähner, J.: Comparing different metaheuristics for model selection in a supervised learning classifier system. In: *Genetic and Evolutionary Computation Conference Companion (GECCO 2022 Companion)* (2022). <https://doi.org/10.1145/3520304.3529015>
31. Yeh, I.C.: Modeling of strength of high-performance concrete using artificial neural networks. *Cem. Concr. Res.* **28**(12), 1797–1808 (1998). <https://doi.org/10/dxm5c2>