Separating rule discovery and global solution composition in a learning classifier system

Michael Heider, Helena Stegherr, Jonathan Wurth, Roman Sraj, Jörg Hähner

Angaben zur Veröffentlichung / Publication details:

Heider, Michael, Helena Stegherr, Jonathan Wurth, Roman Sraj, and Jörg Hähner. 2022. "Separating rule discovery and global solution composition in a learning classifier system." Proceedings of the Genetic and Evolutionary Computation Conference Companion, 248–51. https://doi.org/10.1145/3520304.3529014.

Separating Rule Discovery and Global Solution Composition in a Learning Classifier System

Michael Heider
Michael.Heider@uni-a.de
Universität Augsburg
Augsburg, Germany

Helena Stegherr
Helena.Stegherr@uni-a.de
Universität Augsburg
Augsburg, Germany

Jonathan Wurth
Jonathan.Wurth@uni-a.de
Universität Augsburg
Augsburg, Germany

Roman Sraj
Roman.Sraj@uni-a.de
Universität Augsburg
Augsburg, Germany

Jörg Hähner
Joerg.Haehner@uni-a.de
Universität Augsburg
Augsburg, Germany

ABSTRACT

While utilization of digital agents to support crucial decision making is increasing, trust in suggestions made by these agents is hard to achieve. However, it is essential to profit from their application, resulting in a need for explanations for both the decision making process and the model. For many systems, such as common black-box models, achieving at least some explainability requires complex post-processing, while other systems profit from being, to a reasonable extent, inherently interpretable. We propose a rule-based learning system specifically conceptualised and, thus, especially suited for these scenarios. Its models are inherently transparent and easily interpretable by design. One key innovation of our system is that the rules' conditions and which rules compose a problem's solution are evolved separately. We utilise independent rule fitnesses which allows users to specifically tailor their model structure to fit the given requirements for explainability.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by regression**; **Rule learning**.

KEYWORDS

rule-based learning, learning classifier systems, evolutionary machine learning, interpretable models, explainable AI

1 INTRODUCTION

With increasing automation and digitisation, interaction between humans and trained digital agents becomes more widespread. Such socio-technical systems are for example encountered in smart factory settings. Here, human stakeholders are dependent on recommendations made or decisions taken by an agent, e.g. for (re-) configuring a machine. However, at the moment complex learning tasks can rarely be solved perfectly—often because the available data for training is rather limited, e.g. small sample, large imbalances. This creates a distrust in the entire model among stakeholders, supposedly even if only edge cases were affected. Hard to understand models even exacerbate this issue. The cases of poor performance are rarely easily identifiable and even for good performance on test data, stakeholders often doubt the ability of models with a low transparency.

A common approach to increase stakeholder trust in predictions is explaining the training and prediction processes themselves or the model in its entirety. With increasing model complexity, which is needed for difficult learning tasks, explaining the model or its predictions is less straightforward, leading to some types of models, e.g. rule-based learners, being favoured for these situations, sometimes over better performing ones. Learning Classifier Systems (LCSs) are a family of rule-based learning algorithms that inherently allow application in the aforedescribed settings. [8]

LCS models are composed of a finite set of if-then rules for which the conditions are optimized using a—typically evolutionary—metaheuristic [20]. Rules contain submodels of the problem that apply to certain areas of the feature space. These submodels are comparatively simpler than models for the full problem, thus, increasing their comprehensibility by humans. Most LCSs follow the Michigan-style (a single set whose rules are adapted over time), featuring strong online learning capabilities and being employed to solve all major machine learning tasks. However, these types of systems typically construct (and keep in their population) many more, sometimes suboptimal, rules than would be required to solve the problem at hand. A common approach is, therefore, the reduction of the population to the essential rules after training has concluded, using so called *compaction* techniques [15, 17]. The main other style LCSs follow is the Pittsburgh-style. Here, a population of sets of rules is evolved over time to solve learning tasks. As a set of rules is assigned a combined fitness, rather than individual fitnesses for rules, optimal positioning/selection of rules is more

difficult to achieve for the optimizer, especially as usually multiple changes to the set are performed per optimization step. Suboptimal positioning does not necessarily substantially harm system performance. However, the importance of improving it increases when explanations for rule conditions or the training process are requested. In general, the learning process is envisioned to create an "accurate and maximally general" [18], "maximally accurate and maximally general" [4] or "maximally general as well as accurate" [10] set of rules. Existing LCS rarely specifically target explanations or transparency beyond the non formally specified requirement of generality, although they are still building interpretable models.

In this paper we present a new LCS algorithm that is specifically designed to evolve both individual rules as well as the global problem solution (rule set), with performance as well as explainability considered during optimization. To facilitate this we separate the optimization of rule conditions (find partitions of the feature space for which a submodel of the given type can be fit well) from optimizing a problem solution using these rules (cf. Section 3). We name our system the *Supervised Rule-based Learning System (SupRB)*, as it follows the same general goals as the Pittsburgh-style SupRB-1 [9].

2 RELATED WORK

The most well known LCSs are XCS and its derivatives [20]. While XCS was originally designed for reinforcement learning tasks, it has (with some extensions) been applied in all of the three major learning settings [20]. One of these extensions is the usage of interval-based matching functions rather than binary ones to operate in real-valued environments [21]. To solve supervised function approximation tasks, XCS' constant predicted payoff was replaced with a linear function forming the original XCSF [23]. The linear model and the interval-based matching functions have later on been substituted with various more complex options [3, 13]. These are, however, sacrificing transparency for a stronger predictive performance.

Two approaches to reach *explainability* and its related and relevant concepts of *interpretability* and *transparancy* and thus, ultimately, *understandability* (in this paper we refer to those concepts under a broader umbrella of *explainability* in the spirit of explainable artificial intelligence as a whole), must be distinguished [2]: By intentionally designing *transparent models*, the model structure itself can be used for its comprehension and the interpretation of the decisions made. For other models, *post-hoc methods* that operate through visualisation, transformation of complex black-box models into transparent models and other, often model-specific, techniques, need to be utilised. Like other rule-based learning systems, LCSs can, in general, be seen as transparent/interpretable by design. They also relate to human behaviour naturally [2]. There are, however, some limitations that arise primarily through the encoding of variables, the size of the rule set and the complexity of individual rules.

In LCSs these limitations are typically controlled by design. The variables themselves are problem dependent, so overall influence is limited, but using easy to understand matching functions that allow to follow the implications for decision boundaries in the feature space improves model transparency. However, if the variable/feature itself is highly complex, human interpretation is always limited. Rule complexity is likewise chosen by using a fitting submodel to balance users' transparency requirements with predictive power. Additionally, human understanding can be improved post-hoc by employing a variety of different visualisation techniques for classifiers [14, 16, 19].

In contrast to these generally applicable solutions, handling rule set size is approached differently depending on the LCS(-style). Pittsburgh-style LCSs can control set size directly via their fitness function. For example, GAssist [1] can use the minimum description length in combination with accuracy to form a single objective fitness and additionally apply a penalty on individuals' fitnesses when the rule set size falls below a predefined threshold. In Michigan-style systems, where fitness refers to a rule rather than a rule set and training benefits from larger than necessary populations, similar mechanisms are not available. Instead, *compaction* mechanisms have been designed [6, 24]. After training is completed, they remove redundant or incorrect rules from the population. Ideally, the rule set size decreases without a negative effect on predictive power. This has first been demonstrated [22] on the Wisconsin Breast Cancer dataset and further advanced until the issue was considered solved by Tan et al. [17]. Recently, Liu et al. [15] have proposed new compaction techniques and demonstrated their improvements over existing methods on a variety of boolean benchmarking and three real world problems.

There are some hybrid rule-based learning systems which combine explicit Michigan- and Pittsburgh-style phases for improving explainability by reducing the number of rules [5, 7, 11, 12]. Most utilise the same evolutionary algorithm for both phases, often some multi-objective evolutionary algorithm to find a proper balance between the number of rules and the accuracy. Furthermore, the two phases can be applied subsequently [5], nested [12] or cyclic, where both phases are executed several times [7].

3 THE SUPERVISED RULE-BASED LEARNING SYSTEM

The main idea of SupRB is to optimize rule conditions independently of other rules, discovering a diverse pool of well proportioned rules and then use another optimization process to select a good subset of all available rules to find good solutions to the learning problem. By separating these optimization processes both can include multiple objectives to improve explainability of the LCS model, while still targeting overall performance, e.g. rules should encompass large feature spaces but be positioned to allow a well fitted submodel and solutions should be composed of only few rules while still minimizing prediction error. A Python implementation of SupRB is available on GitHub[1].

For unknown problems, it is hard to estimate how many rules will likely need to be discovered before a good subset can be selected. Therefore, we alternate between phases of discovering new rules and combining rules from the pool of discovered and fitted rules. The expectation is that we can find a good solution with fewer submodel fittings than with conservative estimates of needed rules, while still being able to find such a solution if the number of

[1] https://github.com/heidmic/suprb https://doi.org/10.5281/zenodo.6460701

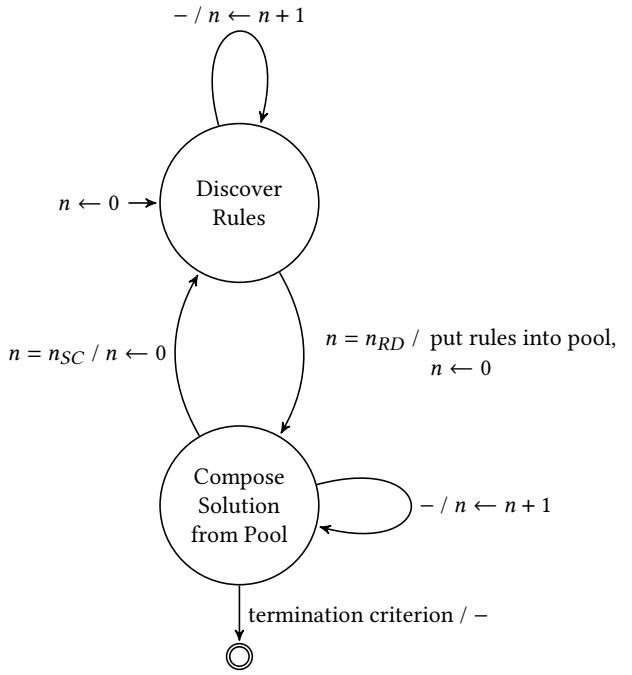

Figure 1: Rule discovery and solution composition phases in SupRB. n_{SC} denotes the number of steps the solution creating/composing optimizer should undertake, while n_{RD} references the number of steps performed within rule discovery.

rules was underestimated. Note that rules added to the pool remain unchanged and will not be removed throughout the training process. Another advantage of alternating between phases is that we can use information from the solution composition phase to steer subsequent rule discoveries towards exploring regions where no or ill-placed rules are found. The overall process is illustrated in the form of a statemachine in Figure 1. The number of optimization steps performed within each phase can be varied, which can impact overall convergence time but does not affect solution strength.

As deriving insights into decisions (cf. Section 2) is a central aspect of SupRB, its model is deliberately kept as simple as possible:

(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.

3.1 Rule Discovery

To discover a new rule for the pool we use an evolution strategy (ES). Note that, in contrast to the hybrid systems described at the end of Section 2, this rule discovery approach can not be considered a Michigan-style phase, especially as new rules are evolved

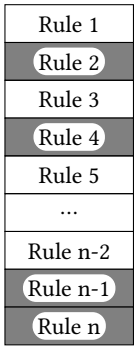

Figure 2: Example global problem solution for a pool of size n. Selected rules are highlighted. In binary notation (on which the optimizer operates) this individual corresponds to 01010 . . . 011**.**

one at a time. Its initial individual is placed around a randomly selected training example, prioritizing examples that have a high in-sample error in the current (intermediate) global solution. Then we use a mutation operator without adaptation that samples a halfnormal distribution twice and moves the upper and lower bound further from the center, according to the respective values, to create λ children. From these, we replace the parent with the individual that has the highest fitness based on its in-sample error and the matched feature space volume. Specifically, the fitness is calculated as

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

with

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

and

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

Its base form (cf. eq. (1)) was adapted from [25], where it was used in a feature selection context, similarly combining two objectives. The Pseudo-Accuracy (PACC) squashes the Mean Squared Error (MSE) of a rule's prediction into a $(0, 1]$ range, while the volume share $V \in [0, 1]$ of its bounds is used as a generality measure. The parameter $\beta = 2$ controls the slope of the PACC and α weighs the importance of o_1 against o_2. Maximizing both objectives hence corresponds to generating rules that have minimal error and are maximally general. A special form of plus-selection is used in the ES, which simultaneously controls the number of iterations: for every iteration, the best of λ children is saved as an elitist and compared with all elitists from previous iterations. If the elitist from δ iterations before is better than all subsequent elitists, the optimization process is stopped and this specific elitist is added to the pool. This procedure to discover a new rule for the pool is performed multiple times before this phase ends. As this optimizer is not meant to find a globally optimal rule but rather fill a multitude of niches with optimally placed rules, independent evolution is advantageous.

3.2 Solution Creation

After new rules have been discovered, a genetic algorithm (GA) selects rules from the pool to form a new solution candidate (a set of rules). Solution candidates are represented as bit strings, signalling whether a rule from the pool is part of the candidate (cf. Figure 2). The GA is configured to use tournament selection and combine two candidate solutions using n-point crossover with a crossover probability. Afterwards, each individual bit of the children is flipped with a probability given by the mutation rate. The candidate fitness is similarly based on eq. (1), using the candidate's in-sample MSE and *complexity*, i.e. the number of rules selected, as first and second objective, respectively. A certain number of elitist solutions from the previous population is additionally copied to the new population without modification. Note that individuals in the GA always form 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, in contrast to typical Pittsburgh-style systems.

4 CONCLUSION

We presented a new Learning Classifier System (LCS) that separates the process of rule discovery from the composition of rules to form problem solutions. This system performs supervised batch-learning and is therefore called the Supervised Rule-based Learning System (SupRB). It utilizes a population-based optimizer (genetic algorithm) whose individuals transcribe which rules from the pool of discovered and locally optimized rules are part of a solution. In contrast to many Pittsburgh-style approaches, which also evolve populations of rule sets, the rules from the pool are always and automatically available to all individuals. Optimization of individuals combines a fitness pressure for low errors and low complexities (number of rules). To fill the pool with rules, we utilized a simplistic ES ($\mu = 1$) that optimizes towards low in-sample errors and high volumes of matched feature space. Note that in contrast to other similar systems, the rule discovery is not done in a Michigan-style phase. Rules are added sequentially in separated evolutionary processes and fitnesses are independent from each other.

The primary motivation for our new approach at creating LCS models was to achieve a greater and more direct control over rule set sizes and matching functions, and thus the overall model structure. Finding a good model structure is also known as the model selection problem. Ultimately, this leads to more interpretable models that make providing explanations for both the model itself, as well as its predictions, easier.

REFERENCES

[1] Jaume Bacardit. 2004. *Pittsburgh genetics-based machine learning in the data mining era: representations, generalization, and run-time.* Ph. D. Dissertation. PhD thesis, Ramon Llull University, Barcelona.

[2] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

[3] Larry Bull and Toby O'Hara. 2002. Accuracy-Based Neuro and Neuro-Fuzzy Classifier Systems. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation* (New York City, New York) *(GECCO'02)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 905–911.

[4] Martin V. Butz, Tim Kovacs, Pier Luca Lanzi, and Stewart W. Wilson. 2004. Toward a Theory of Generalization and Learning in XCS. *IEEE Transactions on Evolutionary Computation* 8, 1 (2 2004), 28–46.

[5] Yung-Hsiang Chan, Tsung-Che Chiang, and Li-Chen Fu. 2010. A two-phase evolutionary algorithm for multiobjective mining of classification rules. In *IEEE Congress on Evolutionary Computation* (2010-07). IEEE. https://doi.org/10.1109/cec.2010.5586523

[6] Phillip William Dixon, Dawid Wolfe Corne, and Martin John Oates. 2003. A Ruleset Reduction Algorithm for the XCS Learning Classifier System. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 20–29. https://doi.org/10.1007/978-3-540-40029-5_2

[7] Dipankar Dutta, Jaya Sil, and Paramartha Dutta. 2020. A bi-phased multi-objective genetic algorithm based classifier. *Expert Systems with Applications* 146 (2020), 113163. https://doi.org/10.1016/j.eswa.2019.113163

[8] Michael Heider, Richard Nordsieck, and Jörg Hähner. 2021. Learning Classifier Systems for Self-Explaining Socio-Technical-Systems. In *Proceedings of LIFE-LIKE 2021 co-located with 2021 Conference on Artificial Life (ALIFE 2021)*, Anthony Stein, Sven Tomforde, Jean Botev, and Peter Lewis (Eds.). http://ceur-ws.org/Vol-3007/

[9] Michael Heider, David Pätzel, and Jörg Hähner. 2020. SupRB: A Supervised Rule-based Learning System for Continuous Problems. (2020). arXiv:2002.10295 [cs.LG]

[10] John H. Holland, Lashon B. Booker, Marco Colombetti, Marco Dorigo, David E. Goldberg, Stephanie Forrest, Rick L. Riolo, Robert E. Smith, Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson. 2000. What Is a Learning Classifier System? In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 3–32. https://doi.org/10.1007/3-540-45027-0_1

[11] Hisao Ishibuchi and Takashi Yamamoto. 2004. Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy Sets and Systems* 141, 1 (2004), 59–88. https://doi.org/10.1016/s0165-0114(03)00114-3

[12] Hisao Ishibuchi, Takashi Yamamoto, and Tomoharu Nakashima. 2005. Hybridization of Fuzzy GBML Approaches for Pattern Classification Problems. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* 35, 2 (2005), 359–365. https://doi.org/10.1109/tsmcb.2004.842257

[13] Pier Luca Lanzi and Daniele Loiacono. 2006. XCSF with Neural Prediction. In *2006 IEEE International Conference on Evolutionary Computation*. 2270–2276. https://doi.org/10.1109/CEC.2006.1688588

[14] Yi Liu, Will N. Browne, and Bing Xue. 2019. Absumption to Complement Subsumption in Learning Classifier Systems. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 410–418. https://doi.org/10.1145/3321707.3321719

[15] Yi Liu, Will N. Browne, and Bing Xue. 2021. A Comparison of Learning Classifier Systems' Rule Compaction Algorithms for Knowledge Visualization. *ACM Transactions on Evolutionary Learning and Optimization* 1, 3 (Aug. 2021), 10:1–10:38. https://doi.org/10/gn8gjt

[16] Yi Liu, Will N. Browne, and Bing Xue. 2021. Visualizations for rule-based machine learning. *Natural Computing* (29 Jan 2021). https://doi.org/10.1007/s11047-020-09840-0

[17] Jie Tan, Jason Moore, and Ryan Urbanowicz. 2013. Rapid Rule Compaction Strategies for Global Knowledge Discovery in a Supervised Learning Classifier System. In *ECAL 2013: The Twelfth European Conference on Artificial Life*. MIT Press, 110–117. https://doi.org/10.7551/978-0-262-31709-2-ch017

[18] Ryan J. Urbanowicz and Will N. Browne. 2017. *Introduction to Learning Classifier Systems* (1st ed.). Springer Publishing Company, Incorporated.

[19] Ryan J. Urbanowicz, Ambrose Granizo-Mackenzie, and Jason H. Moore. 2012. An Analysis Pipeline with Statistical and Visualization-guided Knowledge Discovery for Michigan-style Learning Classifier Systems. *IEEE Computational Intelligence Magazine* 7, 4 (2012), 35–45. https://doi.org/10.1109/MCI.2012.2215124

[20] Ryan J. Urbanowicz and Jason H. Moore. 2009. Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. *Journal of Artificial Evolution and Applications* 2009 (2009).

[21] Stewart W. Wilson. 2000. Get Real! XCS with Continuous-Valued Inputs. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 209–219. https://doi.org/10.1007/3-540-45027-0_11

[22] Stewart W. Wilson. 2001. Mining Oblique Data with XCS. In *Advances in Learning Classifier Systems*, Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 158–174.

[23] Stewart W. Wilson. 2002. Classifiers that approximate functions. *Natural Computing* 1, 2/3 (2002), 211–234. https://doi.org/10.1023/a:1016535925043

[24] Stewart W. Wilson. 2002. Compact Rulesets from XCSI. In *Advances in Learning Classifier Systems*. Springer Berlin Heidelberg, 197–208. https://doi.org/10.1007/3-540-48104-4_12

[25] Qing Wu, Zheping Ma, Jin Fan, Gang Xu, and Yuanfeng Shen. 2019. A Feature Selection Method Based on Hybrid Improved Binary Quantum Particle Swarm Optimization. *IEEE Access* 7 (2019), 80588–80601. https://doi.org/10/gnxcfb