Association Rule Mining as Knowledge Infusion into the Mechanics of an eXtended Classifier System

1st Damijan Novak
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
damijan.novak@um.si

2nd Domen Verber
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
domen.verber@um.si

3rd Iztok Fister
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
iztok.fister@um.si

4th Iztok Fister Jr.
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
iztok.fister1@um.si

***Abstract*—This article uses two well-established research areas: Numerical Association Rule Mining and the eXtended Classifier Systems. With their synergy, an attempt is made to advance the reuse of previously generated associated rules of a given dataset. Additionally, the article investigates how integrating infused rules into the XCS algorithm's population impacts its adaptive capabilities, and how well the learned knowledge transfers to testing dataset (environment) scenarios. This article also explores the novel approach of utilizing any user-selected dataset feature as a prioritized action for the eXtended Classifier System algorithm to adapt to. This adaptability is enabled by incorporating the prioritized action into the reinforcement learning process guided by an evaluation function. Such extended applicability should go beyond traditional classification tasks.**

***Index Terms*—adaptivity, Association Rule Mining, eXtended Classifier System, rule re-usage, reinforcement learning**

I. Introduction

Humanity creates vast amounts of data and stores it for commercial, research, or government purposes [1]. The data increases come on behalf of the constant rise of data sensors present in everyday lives, the technological interconnections of people across the internet (e.g., social networks, business applications, metaverses, etc.), due to the need to help us understand climate change better, due to high increase in (private) space explorations, and due to numerous similar cases. The artificial intelligence techniques that are tackling this data (e.g., big data [2] or large language models [3]) are in a new technological "Renaissance" due to many research breakthroughs. Its impact supports research ranging across a number of domains (e.g., improving the well-being of people through the help of medicine [4]).

This research was funded by the Slovenian Research Agency Research Core Funding No. P2-0057.

However, considerations (or limitations) are always present when processing large amounts of data is needed. Questions arise, such as how many computational resources are needed for processing, how soon they can be processed, or how interpretable the results are. For example, deep neural networks (DNN), when operating at their peak value, can deliver results on-par with the best-performing people in specific domains (e.g., on a grand master level in the popular real-time strategy game of StarCraft 2 [5]). But, they usually require vast processing power, are hard to interpret (i.e., they can be seen as a black-box system [6]), and require much data to train (e.g., to minimize over-fitting [7]). Therefore, much research was done to make them more adaptable and understand better how their approximation properties work [8].

Nevertheless, DNNs are not the only possible solution to the problem. Considerations for simultaneously researching other Machine Learning (ML) methods are also highly relevant [9]. Therefore, this work is positioned in the classical ML algorithms domain (i.e., distinct from DNN). The intention is to provide a complementary contribution with other ML algorithms, facilitating the reuse of knowledge, making the algorithms more adaptable to new data (e.g., in medicine, there is a great need for that [10]), and potentially enhancing their interpretability in the process. Specifically, the article focuses on the ML domains of Learning Classifier Systems (LCS) and Association Rule Mining (ARM). For precision, the investigation includes the sub-domains of eXtended Classifier Systems (XCS) and Numerical ARM (NARM).

The LCSs are ML algorithms that integrate Reinforcement Learning (RL) with the evolution strategies. In other words, these systems use a set of classifiers called rules, which evolve through a combination of RL and (usually) a genetic algorithm, adapting to the changing environment [11]. The research in the LCS domain spans several decades already. It includes many essential techniques that deal with online (i.e., maintaining real-time responsiveness) adaptivity or offline (i.e., over more extended periods) optimization in learning scenarios (online are now considered as Michigan, and offline, as Pittsburgh classifier systems) [12]. For an interested reader who wants to delve deeper into the background of LCS systems and their history, it is warmly recommended to check the works of [13] and [14]. This work used the XCS group of algorithms, known for being online adaptive and evolving accurate, maximally generalized classifiers [15].

ARM is a data mining method aimed at discovering relationships between objects in transaction databases [16]. ARM methods are efficient for providing new knowledge, representing mathematical implications consisting of two parts (i.e., an antecedent and a consequence) [17]. The problem was initially defined by Agrawal et al. [18]. Since this definition, many algorithms for mining association rules have been proposed in past years. At first, these algorithms were based on deterministic methods [19], while, nowadays, many algorithms are founded on stochastic population-based methods [20]. Interestingly, classical ARM methods can work only on datasets with discrete attributes. Still, modern methods are conceived more universally, and, thus, they allow working with mixed types of attributes in datasets (i.e., numerical and discrete). These methods belong to a class of NARM [17], and are based primarily on stochastic population-based nature-inspired algorithms [21] that present efficient methods for exploring huge search spaces.

Three aspects of data utilization were addressed when integrating NARM and XCS (NARM-XCS). Firstly, the study focuses on reusing ARM results, which can be computationally expensive to produce. Secondly, it involves connecting the knowledge embedded in the ARM data with executable actions through RL mechanisms. Note that, in our work, any feature (attribute) from the dataset can serve as a prioritized action for which the XCS learns associations between the environment input and the proper action output. Importantly, this action doesn't necessarily have to be a class attribute from a classification dataset. The XCS algorithm adapts its rules to learn and predict the optimal actions based on the chosen feature, making it versatile and applicable to scenarios beyond traditional classification tasks. Thirdly, emphasis is placed on augmenting the adaptability of the NARM-XCS algorithm to transition seamlessly and accommodate the new environmental state when presented with new sensory input data. Notably, this process eliminates the need for computationally expensive retraining of the whole model, distinguishing it from certain other algorithms.

The structure of the article is as follows. Related works are presented in the second Section. In the third Section, the NARM and XCS algorithms are described in more detail, and their connection is provided in the form of an NARM-XCS architecture. Section four presents an experimental evaluation of the NARM-XCS architecture. The results and a short discussion are given in Section five. The article ends with future work directions in Section six.

II. Related works

In the work of [22], the authors investigated the evolution of ARM rules with LCS from streams of unlabeled examples. Their primary goal was to extract the categorical and quantitative association rules in online mode, and to address the concept drift (i.e., when the underlying patterns in data change over time) in new, challenging real-world problems. This work could be seen as an inspiration for our work, but with a few key differences. The study's authors focused on generating the rules and analyzing these discovered (interesting, as they put it) rules. In contrast, our study is focused on prioritized actions, which influence the direction of the attribute space the XCS algorithm should take (or adapt to). Also, in our work, the XCS algorithm starts from the outset of previously mined ARM rules, and uses them as an initial foundation of knowledge. Therefore, the start of the algorithm processing is different.

The adaptive mechanics are well-documented mechanisms in the Evolutionary Computation domain (e.g., the self-adapting algorithms which, by themselves, without external help, control the transmission function between the parent and the offspring population [23]). For instance, the authors in [24] enhanced the XCS algorithm by introducing an adaptive mapping mechanism that evolves solutions-focused actions dynamically with the most significant reward returns. Our work differs from this work in a way that utilizing any dataset feature is a less general, but more robust approach, that can be used to improve the performance of RL algorithms by focusing on specific features.

III. Association Rule Infusion in Extended Classifier System

The main components of the proposed architecture are described in this Section. Specifically, the primary operational mechanism of NARM and XCS, and how they connect. Then, a graphical overview of the association rule infusion in XCS architecture is presented. The limitations and future extensions are also discussed, due to the many opportunities identified during the development phase.

A. Numerical Association Rule mining

ARM aims to discover the relations between attributes hidden in transaction databases. Initially, most algorithms for mining association rules operate deterministically, focusing on datasets consisting of discrete attributes only. With the vast development in domains of Evolutionary Algorithms [21] and Swarm Intelligence-based algorithms [25], universal algorithms for mining mixed types of attributes (i.e., discrete and numerical) have been developed. These algorithms are known under the name NARM [26].

The NARM problem is defined formally as follows: Let us suppose a set of features $\mathcal{O} = \{O_1, \ldots, O_m\}$ and a transaction database Db are given, where the transaction database consists of transaction Tr, while each transaction contains a subset of features $Tr \subseteq \mathcal{O}$. Indeed, each feature O_i for $i = 1, \ldots, m$ is represented as either a set of categorical attributes $O_i^{(cat)} = \{a_{i,1}, \ldots, a_{i,n_i}\}$, where $a_{i,j}$ for $j = 1, \ldots, n_i$ denote discrete values and n_i is the number of attributes, or an interval of real values $O_i^{(num)} \in [lb_i, ub_i]$, where $lb_i \geq LB_i$ and $ub_i \leq UB_i$ designate the lower and upper bounds of definite interval inside the domain of feasible values for the feature (i.e., the interval $[LB_i, UB_j]$). Thus, the relation must be satisfied as follows $0 < ub_i - lb_i \leq \Delta_i$, where Δ_i denotes the maximum interval size. By these definitions, an association rule is defined as an implication [27]:

$$X \Rightarrow Y, \tag{1}$$

where $X \subset O$, $Y \subset O$, and $X \cap Y = \emptyset$. The quality of the association rules is typically evaluated using the following two measures [27]:

$$\mathit{supp}(X \Rightarrow Y) = \frac{n(X \cup Y)}{N}, \tag{2}$$

and

$$\mathit{conf}(X \Rightarrow Y) = \frac{n(X \cup Y)}{n(X)}, \tag{3}$$

where $\mathit{supp}(X \Rightarrow Y) \geq S_{min}$ denotes the support and $\mathit{conf}(X \Rightarrow Y) \geq C_{min}$ the confidence of the association rule $X \Rightarrow Y$. In Eq. (2), the variable N represents the number of transactions in the transaction database Db, and $n(.)$ is the number of repetitions of the particular rule $X \Rightarrow Y$ within Db. Furthermore, C_{min} denotes minimum confidence and S_{min} minimum support, determining that only those association rules with confidence and support higher than C_{min} and S_{min} are considered, respectively.

Although many other quality measures have been defined recently, the measures mentioned above are elementary enough that their applications are the most widespread.

B. eXtended Classifier System

XCSs are rule-based evolutionary online learning systems [28]. They are part of the larger LCS domain. They are also a model-free system. That means the XCS doesn't attempt to comprehensively represent the relationships or dynamics within the data (i.e., no detailed environment model is constructed). Instead, it focuses on learning associations between inputs and outputs from the data [29]. Learned associations (knowledge) are kept in a so-called population, which is a set of classifiers.

A classifier is a relatively simple concept expressed as a form of an algorithmic IF sentence, and consists of a condition and an action. The classifier functions so that if the specified condition is met, the action that the classifier is propagating can be executed or used during the XCS main loop execution run (e.g., when predictions about available actions that can be executed in an environment are generated). A condition also supports the concept of a 'don't care' (#) mechanism. This allows classifiers to generalize over specific inputs by treating parts of conditions as 'don't care'. With this mechanism, the classifier can match a broader range of input patterns (sensory inputs), providing flexibility in learning. The classifier also supports some additional parameters that define it further. The main additional parameters are usually the prediction estimate (if the classifier matches an environment input, this is the payoff we can expect), prediction error (estimates of the errors for the given prediction), experience (a counter of how many times the classifier was belonging to the set of propagated actions), and fitness (fitness tells how accurate the prediction estimate is).

The XCS modus operandi can be summarized as follows. It uses three sets (population, match set, and action set, all comprising classifiers) and operates within a main execution loop. The iterative behavior can be outlined in the following short steps. First, the environment is observed, and the sensory input is received. In our case, the environment sensory input is one encoded line (or an attribute and its value when an evaluation function is involved) of a training dataset line or of a testing dataset—depending on the XCS that is running in the training (active learning) or testing (i.e., inferencing actions) phase. Second, the algorithm generates a match set by selecting classifiers that match the current input. Third, using this match set, XCS produces predictions for possible actions. The fourth step involves the selection of an action based on these predictions. Following the execution of the chosen action, the algorithm receives feedback from the environment, allowing it to adapt and evolve the classifiers in its population. This adaptive cycle repeats iteratively, enabling XCS to learn and refine its rules over time.

C. NARM-XCS Architecture

Fig. 1 presents the NARM-XCS architecture and how the components are connected. There are four main steps involved in the operation of the architecture. The first is the data preparation and initialization, the second is the generation and processing of NARM rules, the third is for selecting and using an encoder on the rules and the dataset lines, and the fourth step is infusing the encoded rules into the XCS algorithm. The encoded rules are then evolved using the environment, RL components, and the evaluation function, which guides the search for the prioritized action.

A crucial aspect of the first two steps is that the dataset's attributes are used directly as actions by the XCS algorithm. When the ARM rules are integrated into the XCS population, the consequents of these rules become actions. The values of the actions are also kept, although they are not adapted during the main execution loop (i.e., reserved for future work). In the third step, an encoder has to be used to encode the dataset lines and rules into a format suitable for processing by the XCS. In our experiment, a basic binary encoder was used, with more sophisticated encoders being a definitive improvement option to try. The binary encoder encodes numerical data into a set of binary values by calculating the bin number for each attribute value, converting the bin number to a binary string with the appropriate number of bits, and adding each bit to an encoded list. The encoder takes as input a dataset or the NARM rules, a list of minimum and maximum values for each attribute (calculated by parsing the dataset or the rules), the number

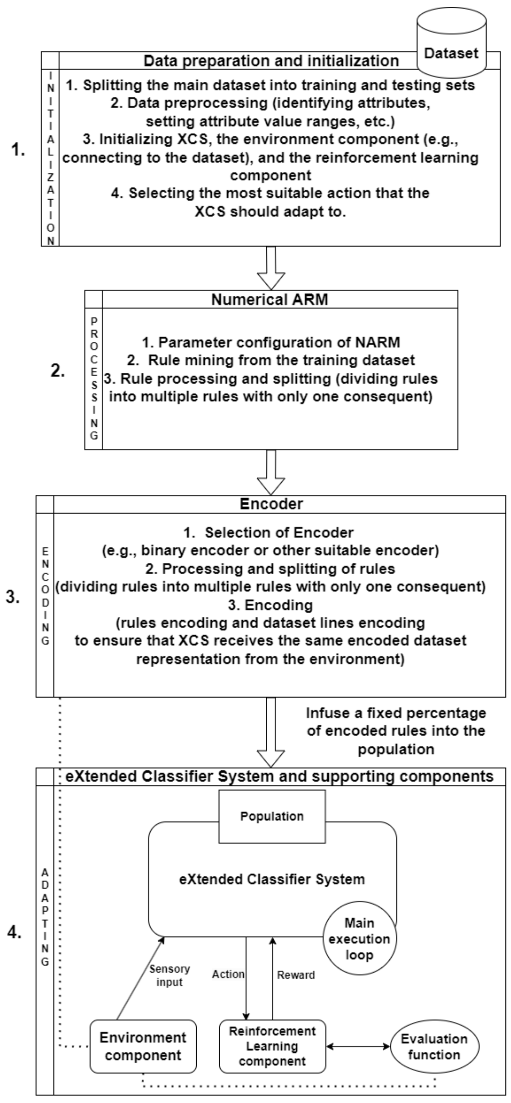
Dataset
Data preparation and initialization
INITIALIZATION
1.
1. Splitting the main dataset into training and testing sets
2. Data preprocessing (identifying attributes, setting attribute value ranges, etc.)
3. Initializing XCS, the environment component (e.g., connecting to the dataset), and the reinforcement learning component
4. Selecting the most suitable action that the XCS should adapt to.
Numerical ARM
PROCESSING
2.
1. Parameter configuration of NARM
2. Rule mining from the training dataset
3. Rule processing and splitting (dividing rules into multiple rules with only one consequent)
Encoder
ENCODING
3.
1. Selection of Encoder (e.g., binary encoder or other suitable encoder)
2. Processing and splitting of rules (dividing rules into multiple rules with only one consequent)
3. Encoding (rules encoding and dataset lines encoding to ensure that XCS receives the same encoded dataset representation from the environment)
Infuse a fixed percentage of encoded rules into the population
eXtended Classifier System and supporting components
ADAPTING
4.
Population
eXtended Classifier System
Main execution loop
Sensory input
Action
Reward
Environment component
Reinforcement Learning component
Evaluation function

Fig. 1. Key components and core mechanisms of the NARM-XCS architecture.

of bins, and an encoding attribute setting (only used when encoding rules). It then encodes each attribute in the dataset or the rule set by calculating the bin the attribute value falls into and then converting the bin to a binary string with the appropriate number of bits.

IV. Experiment

With this experiment, a preliminary investigation into an infusion of the rules in the population of the XCS algorithm was performed to test if the learning performance of the XCS algorithm was improved. The study also aimed to assess the outcomes of the training section, and how stable the reward received was during the subsequent testing phase of the experiment. The experiment was run on an interval of 0 to 100 percent of encoded rules used in the XCS population, with a step of 10 percent. The training and testing runs were done five times for each step percentage, and the reward average was calculated. Only one pass of the dataset was allowed for each run (i.e., each instance from the training and testing dataset was used only once during the run). Each run's reward was calculated by dividing the cumulative reward gathered during the run by the number of instances in the dataset.

So as not to interfere with the repeatability of the experiment runs, the experiment was designed in such a way that the NARM rule generation was done only once. Therefore, when the rules were generated, the XCS population initializations were completed with the same rules (i.e., 1,329 and 1,120 rules) for all the different experiment settings and all the repeated runs. Therefore, the results and differences in rewards that were received only reflected the operation of the XCS algorithm using the NARM rules in its population initialization, and were not the result of possible randomness when uARMSolver [30] was processing (i.e., the DE that the uARMSolver was using can produce slightly different rules each time, because of its non-fully deterministic behavior). The XCS used in the experiment was implemented in its basic form [31], so any additional enhancements wouldn't influence the experimental results. The settings of the NARM hyperparameters of uARMSolver and of the XCS hyperparameters can be observed in Table I and Table II.

In this experiment, the number of bins in the binary encoder was set to two, and the encoding attribute setting was used to calculate the average value of the rule values. In the fourth step, the XCS main execution loop iterated over the dataset until the termination criteria were met (e.g., the end of the dataset), adapting the rules to the new sensory input.

The reward was calculated by a basic evaluation function, designed to provide a positive reward of 0.2 if the XCS selected the action per the prioritized action, and a negative reward of -0.1 if the XCS didn't choose the proper action. In a subsequent experiment, a positive reward of 1.0 was added (i.e., enhancing the evaluation function) when the encoded XCS selected an action and its action value matched the corresponding encoded attribute and its value from the testing dataset. This subsequent experiment aimed to better understand how NARM-XCS behavior changed, and how its adaptation was affected. Important note: The XCS doesn't include mechanisms to change the values of actions during the main execution loop, but it decides based on a random value selection between the minimum and maximum possible value of an attribute when initializing the population. Therefore, this insight is likely valuable for (our) future studies.

Two classification datasets from the UCI ML repository were used for this study. The first dataset is called Wine [32], which holds 178 instances and has 13 features, and the feature type is comprised of integer and real values. The second dataset is Statlog (Shuttle) [33], which holds 58,000 instances, has eight features, and comprises integer values. The data instances were split randomly into the training and testing samples in the ratio of 80:20. Specifically, the Wine dataset was divided into 142 training instances and 36 testing instances, while Statlog (Shuttle) was divided into 46,400 training instances and 11,600 testing instances. The Wine training instances resulted in the creation of 317 NARM rules,

Parameters	Values
Algorithm	Differential Evolution
NP	100
nFes	1000
F	0.5
CR	0.9

TABLE I
uARMSOLVER HYPERPARAMETERS

Parameters	Values
N (population size)	500
alfa	0.1
beta	0.01
gamma	0.71
delta	0.1
θ_{GA}	25
ε_0	10
θ_{del}	20
ν	5
χ	0.5
μ	0.01
θ_{sub}	20
$P\#$ (probability of using #)	0.33
doGASubsumption	true
doActionSetSubsumption	true

TABLE II
XCS HYPERPARAMETERS

which, when processed, would only have one attribute as a consequent, resulting in a total of 1,329 rules (the XCS algorithm then used these rules). On the other hand, the Statlog (Shuttle) training dataset instances produced 307 rules altogether, and, in the processed form, the total count was 1,120 rules.

For the Wine dataset, a Class feature was used as a prioritized action for the XCS to consider in the evaluation function. In contrast, a randomly selected feature (i.e., Attribute3) was used for Statlog (Shuttle).

V. RESULTS AND DISCUSSION

The results provided in the graphs show the average reward collected during the training and testing runs. Figures 2 and 3 present the results for both datasets with the basic evaluation function. The results indicate that the training and testing rewards rose steadily on the smaller Wine dataset, as more encoded rules were infused into the population. At 100 percent of the infused rules, the XCS reached the average reward of 0.186. The larger dataset of Statlog (Shuttle), on the other hand, reached the convergence of the rewards (approx. 0.188) for both training and testing of rewards much sooner (i.e., before the 40 percent mark). The results, therefore, confirm that the more rules that are infused, the higher the gathered reward is, and the faster convergence towards the top average reward for the XCS algorithm (i.e., only one dataset pass was used in the experiment, and the XCS didn't reach the top reward of 0.188 with lower percentages, indicating the convergence was slower with a lower infusion of the rules into the population). In Fig. 4, the training rewards for the enhanced evaluation function showed small increases with a higher percentage of infused rules, but the testing rewards did

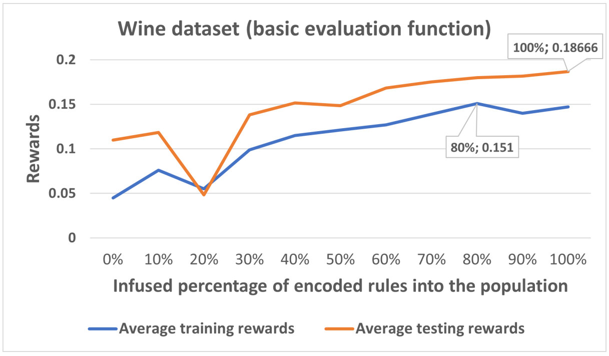

Fig. 2. Rewards of training and testing datasets evaluated using the Wine dataset with a basic evaluation function.

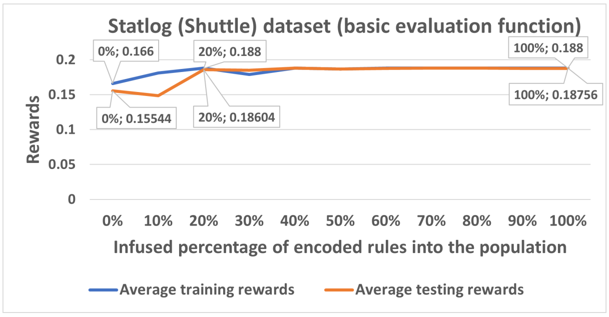

Fig. 3. Rewards of training and testing datasets evaluated using the Statlog (Shuttle) dataset with a basic evaluation function.

not follow this trend. In Fig. 5, the training reward for the enhanced evaluation function remained static at zero, while the testing rewards showed clearly (chaotic) negative trends. The results aligned with our anticipated behavior because the XCS doesn't have the mechanisms to adapt the values of actions at this point.

VI. CONCLUSION AND FUTURE WORK

The following contributions were achieved with this work:

- Bridging the gap between previously generated ARM rules and the proper action-taking of XCS in the environment (e.g., priming for rare events).
- Fast adaptation of the rules with new sensory input data (i.e., can be used to adapt to the specific sub-domains).
- The first steps taken with our work were not only to choose a proper action, but also, in the future, the proper

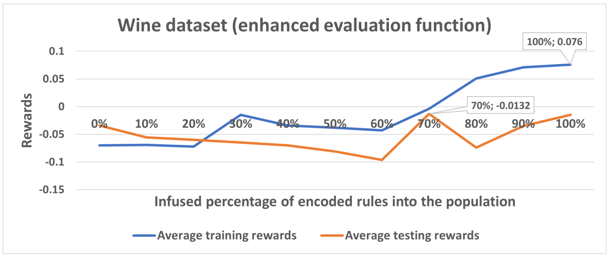

Fig. 4. Rewards of training and testing datasets evaluated using the Wine dataset with an enhanced evaluation function.

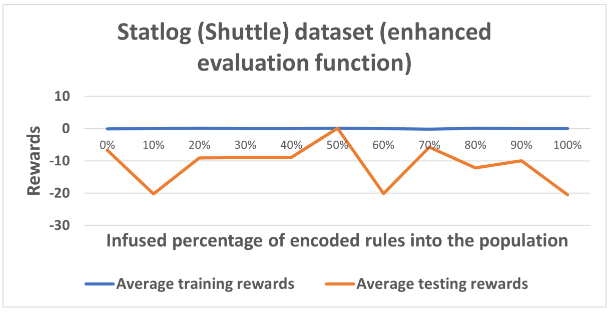

Fig. 5. Rewards of training and testing datasets evaluated using the Statlog (Shuttle) dataset with an enhanced evaluation function.

value for that action (e.g., connecting small subsets of data (rules) to exceptional behavior (actions) [34]).

In future work, we would like to address the possibility of missing attribute input values in a dataset, incorporating advanced mechanisms for adapting the action values during the main execution loop of the XCS algorithm, filtering the rules based on their fitness thresholds (i.e., keeping only the rules that are above the specified threshold quality), an in-depth analysis of all the components (including an ablation study and a time analysis), the usage of more sophisticated encoders (e.g., K-means encoder), to touch on the subject of the explainable nature of the rules of XCS, and improved evaluation functions. The main idea is first to bring the ARM-XCS architecture to peak performance, and then transition to the problem-specific operations (e.g., advancing state-of-the-art game agents operating in online mode with lots of data to process) in complex environments (e.g., environments with partial or incomplete (data) observability).

References

[1] M. N. I. Sarker, M. Wu, and M. A. Hossin, "Smart governance through bigdata: Digital transformation of public agencies," In 2018 international conference on artificial intelligence and big data (ICAIBD), IEEE, pp. 62-70, May 2018.

[2] Y. Duan, J. S. Edwards, and Y. K. Dwivedi, "Artificial intelligence for decision making in the era of Big Data–evolution, challenges and research agenda," International journal of information management, vol. 48, pp. 63-71, 2019.

[3] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, and D. S. W. Ting, "Large language models in medicine," Nature medicine, vol. 29(8), pp. 1930-1940, 2023.

[4] T. Hulsen, S. S. Jamuar, A. R. Moody, J. H. Karnes, O. Varga, S. Hedensted, and E. F. McKinney, "From big data to precision medicine," Frontiers in medicine, vol. 6(34), 2019.

[5] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," Nature, vol. 575(7782), pp. 350-354, 2019.

[6] I. H. Sarker, "Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions," SN Computer Science, vol. 2(6):420, pp. 1-20, 2021.

[7] M. Kim, J. Yun, Y. Cho, K. Shin, R. Jang, H. J. Bae, and N. Kim, "Deep learning in medical imaging," Neurospine, vol. 16(4):657, pp. 657–668, 2019.

[8] Z. Cai, J. Chen, and M. Liu, "Self-adaptive deep neural network: Numerical approximation to functions and PDEs," Journal of Computational Physics, vol. 455: 111021, 2022.

[9] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," arXiv preprint arXiv:2007.05558, 2020.

[10] A. M. Chekroud, M. Hawrilenko, H. Loho, J. Bondar, R. Gueorguieva, A. Hasan, et al., "Illusory generalizability of clinical prediction models," Science, vol. 383, pp. 164–167, 2024.

[11] M. R. Karlsen, and S. Moschoyiannis, "Evolution of control with learning classifier systems," Applied network science, vol. 3, pp. 1-36, 2018.

[12] P. L. Lanzi, "Learning classifier systems: then and now," Evolutionary Intelligence, vol. 1, pp. 63-82, 2008.

[13] L. Bull, "A brief history of learning classifier systems: from CS-1 to XCS and its variants," Evolutionary Intelligence, vol. 9, pp. 55-70, 2015.

[14] R. J. Urbanowicz, and J. H. Moore, "Learning classifier systems: a complete introduction, review, and roadmap," Journal of Artificial Evolution and Applications, 2009.

[15] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson, "Toward a theory of generalization and learning in XCS," IEEE transactions on evolutionary computation, vol. 8(1), pp. 28-46, 2004.

[16] X. Wu, V. Kumar, Q. J Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J McLachlan, A. Ng, B. Liu, P. S. Yu, "Top 10 algorithms in data mining," Knowledge and information systems, vol. 14(1), pp. 1-37, 2008.

[17] I. J Fister and I. Fister, "A brief overview of swarm intelligence-based algorithms for numerical association rule mining," arXiv preprint arXiv:2010.15524, 2020.

[18] R. Agrawal, T. Imieliński and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," In 1993 ACM SIGMOD International Conference on Management of Data, ACM, pp. 207-216, 1993.

[19] C. Borgelt, "An Implementation of the FP-Growth Algorithm," In 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations, ACM, pp. 1-5, 2005.

[20] I. J Fister, A. Iglesias, A. Galvez, J. Del Ser, E. Osaba and I. Fister, "Differential Evolution for Association Rule Mining Using Categorical and Numerical Attributes," In Intelligent Data Engineering and Automated Learning – IDEAL 2018, Springer International Publishing, pp. 79-88, 2018.

[21] A. E. Eiben Agoston E and J. E. Smith, "Introduction to evolutionary computing," Springer-Verlag Berlin, 2015.

[22] A. Orriols-Puig, and J. Casillas, "Evolution of interesting association rules online with learning classifier systems," In International Workshop on Learning Classifier Systems, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 21-37, 2008.

[23] S. Meyer-Nieberg, and H.-G. Beyer, "Self-adaptation in evolutionary algorithms," In Parameter Setting in Evolutionary Algorithms, vol. 54 (of Studies in Computational Intelligence), Heidelberg: Springer Berlin Heidelberg, pp. 19–46, 2007.

[24] M. Nakata, P. L. Lanzi, and K. Takadama, "XCS with adaptive action mapping," In Asia-Pacific Conference on Simulated Evolution and Learning, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 138-147, 2012.

[25] C. Blum and D. Merkle, "Swarm Intelligence: Introduction and Applications," Springer-Verlag Berlin, 2008.

[26] I. Fister, A. Iglesias, A. Galvez and I. J Fister, "Online numerical association rule miner," Neurocomputing, Elsevier, vol. 523, pp. 33-43, 2023.

[27] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules," In 20th int. conf. very large data bases, VLDB, Morgan Kaufmann Publishers Inc., vol. 1215, pp. 487-499, 1994.

[28] N. Fredivianus, H. Prothmann, and H. Schmeck, "XCS revisited: a novel discovery component for the eXtended classifier system," In Asia-Pacific Conference on Simulated Evolution and Learning, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 289-298, 2010.

[29] R. J. Urbanowicz, and W. N. Browne, "Introduction to learning classifier systems," Springer, 2017.

[30] I. Fister and I. Fister Jr. "uarmsolver: A framework for association rule mining," arXiv preprint arXiv:2010.10884 (2020).

[31] M. V. Butz, and S. W. Wilson, "An algorithmic description of XCS," Soft Computing, vol. 6, pp. 144-153, 2002.

[32] S. Aeberhard, and M. Forina, Wine, UCI Machine Learning Repository, 1991, https://doi.org/10.24432/C5PC7J.

[33] Statlog (Shuttle), UCI Machine Learning Repository, https://doi.org/10.24432/C5WS31.

[34] S. Ventura, and J.M. Luna, "Mining Exceptional Relationships Between Patterns," In: Pattern Mining with Evolutionary Algorithms, Springer, Cham, 2016.