An Online Evolutionary Rule Learning Algorithm with Incremental Attribute Discretization

Essam Debie, Kamran Shafi, Kathryn Merrick, and Chris Lokan
School of Engineering and Information Technology, University of New South Wales, Australian Defence Force Academy, Canberra, Australia.
E.Debie@adfa.edu.au, K.Shafi@adfa.edu.au, k.merrick@adfa.edu.au, C.Lokan@adfa.edu.au

***Abstract*—Classification rule induction involves two main processes: finding the optimal conjuncts (attribute intervals or attribute-value pairs) and their combination (disjuncts or rules) to classify different concepts in the data. The evolutionary rule learning approaches employ an evolutionary algorithm, such as a genetic algorithm, to perform both these search operations simultaneously. This approach often leads to significant problems including population bloating and stalled evolutionary search in real-valued attribute problems, especially with higher dimensions. In this paper, we present an online evolutionary rule learning approach referred to as ERL-AID that decouples the above search processes and employs a discretization algorithm that works on the attribute space and a genetic algorithm to combine the discretized attributes into appropriate classification rules. ERL-AID applies a sliding window approach to process inputs in an online fashion. The proposed system is able to produce compact rule sets with competitive performance and could scale to higher dimensions. The experimental results show the competitiveness of our algorithm.**

I. INTRODUCTION

Data classification is a key task in machine learning research which involves automatically categorizing data into distinct concepts using a classification model. A large number of approaches to build classification models have been proposed over the years including those that rely on directly inducing interpretable classification rules from the data. Evolutionary Rule Learning (ERL) systems belong to the rule induction category of machine learning approaches that apply evolutionary algorithms (EA) to build their rule based classification models. The ERL systems have shown competitive performance in comparison to their non-evolutionary counterparts and have been successfully applied to a variety of classification tasks [1], [2]. ERL systems have been proposed with different flavours of EA including those based on genetic algorithm (GA) [3], genetic programming (GP) [1], and evolutionary strategy [4]. A complete taxonomy of such approaches is out of scope of this paper; however, in the related work (Section II), we will expand more upon the GA based ERL which are more relevant to the subject of this paper. A key feature of ERL is their ability to adapt their classification models quickly to changing problem concepts. In this regard, the systems with online learning mechanisms are more suitable and can deal with stream or large-scale data mining problems where multiple data scans are either not possible or computationally too expensive. GA based ERL, also referred to as genetic based machine learning (GMBL) or learning classifier systems (LCS) [5], generally belong to this class of algorithms. These algorithms are generally classified into two main types depending upon how the classification model is represented. Pittsburgh-style ERL [6] evolves a population of classification models (sets of rules) in order to find the best fit model. Whereas, Michigan style ERL [3] evolves a single set of rules and try to search for an optimal rule set. In addition to online learning, dealing with larger classification problems also requires designing scalable learning mechanisms in ERL. The work presented in this paper is motivated under this context.

Rule learning in ERL systems, similar to other rule-based system, rely on two basic operations: finding the optimal conjuncts (attribute intervals or attribute-value pairs) and their combination (disjuncts or rules). Traditionally GA based ERL systems, in specific the Michigan style LCS have used GA for both of these operations. Previous research on the application of such ERL on real-valued large-scale problems [7], [8] has demonstrated the difficulty faced by these systems when dealing with such problems. We attribute this poor performance to not distinguishing between the above mentioned processes and using GA implicitly to carry out both operations simultaneously. Subsequently, we propose a novel ERL system in this paper that focuses on decoupling these two search processes and handling them independently.

The proposed system consists of two main components operating in parallel and cooperating together to find the optimal classification rules. The first component is an incremental discretization algorithm that works on the attribute space to find optimal interval based conjuncts where a list of attribute value pairs (bins) [B] is maintained. The second component is a genetic algorithm that aims at finding the right combination of these conjuncts into optimal rules by using a sliding window [W] of a certain size on the training instances. In a sense, the discretization module enhances GA performance by bootstrapping it with the right sized intervals. Both algorithms are incremental and cooperate to find the maximally general, concise and accurate set of classification rules. The hypothesis is that by allowing GA to focus on one problem only, the search space will be reduced, accordingly computational cost will be reduced and the system will scale to higher dimensions. The discretization component contributes also to the improvement of rule learning by guiding the genetic

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

search dynamically through merging and splitting operators.

By decoupling the discretization problem from the rule learning, not only ERL-AID achieves better discretization results but also the classification solution accuracy increases without a huge increase in the computational cost. The bloating effect, observed in real-valued problems, is alleviated. As a result, the search space is significantly shrunk and the resulting rule set is reduced compared to other techniques that rely on GA for both tasks. Moreover, performing the two tasks concurrently online allows the system to easily adapt to dynamic environments.

The performance of the ERL-AID algorithm is evaluated against other evolutionary learning algorithms using 15 real data sets. As an evaluation criteria, we measure the classification accuracy and the compactness of the generated rule set in terms of the number of rules generated and the number of attributes and conditions used per rule. Using 10-fold cross validation, our algorithm was statistically shown to be the best algorithm in terms of the classification accuracy and the second best in terms of rule compactness.

The remaining sections of this paper is organized as follows: Section 2 briefly reviews the related work. Description of the proposed algorithm ERL-AID is presented in Section 3. Experiments including experimental setup, data sets used, and results and discussions are presented in Section 4. Section 5 summarizes, concludes, and discusses future work.

II. Related Work

Many types of evolutionary rule learning approaches have been proposed in the literature where GAs were used as the primary search mechanism for finding good classification rules. The first model was the Learning Classifier System introduced in 1978 [9]. Since then, many algorithms have been proposed in the literature [3], [10]. Other evolutionary algorithms have been also investigated to the discovery of classification rules such as Genetic Programming [1], Gene expression programming [11]. In this section, we will expand more on the GA-based algorithms as they are more related to our work.

Despite the recent progress in evolutionary rule learning, classification tasks with real-valued attributes remains a non-trivial problem for ERL algorithms, especially in incremental and stream formulations. There is, however, an increasing number of problem domains where both these properties are important. Examples include stock market, intrusion detection, and web mining. An ideal algorithm for such tasks needs to allow for stream incremental learning, avoid potential difficulty with real-valued attributes in the input data, eliminate irrelevant input attributes, and keep the computational complexity of learning low while achieving accurate prediction and adequate generalization.

Some ERLs can only represent rule conditions with categorical (nominal) attributes. If the classification task contains real-avlued attributes, these attributes have to be converted to nominal values using discretization techniques in a preprocessing stage before the ERL is applied. This type of discretization is known in the literature as global discretization. In spite of the solid theoretical foundations that some of these discretization approaches possess, they are not necessarily suitable for stream learning. First, they were developed primarily for batch data analysis and assumes that the training data are all available a priori and discretization can be done separately from rule learning. Second, they are not easily and/or efficiently adjusted for integration with evolutionary rule learning from incrementally arriving data.

Other ERLs can directly represent rule conditions with both categorical and real-valued attributes . These are rather well suited for incremental and stream learning, particularly for low dimensional problems. The ERL is doing a local (embedded) discretization of real values on the fly, since by creating rule conditions such as $30 \leq age \leq 40$ the ERL is generating discrete intervals. These ERLs use GA for dual roles such that it searches at both attribute level and rule level. At the attribute level, the task of EA is to find the optimal interval bounds for each attribute. At the rule level, the task is to find the optimal combinations of attributes intervals that best describe the target concept. However, on the one hand, empirical results have indicated that global discretization methods often produced superior results compared to local embedded methods since the former use the entire value domain of a numeric attribute for discretization, whereas local methods produce intervals that are applied to sub-partitions of the instance space [12]. On the other hand, since these techniques allocate resources to cover the input space in a localized fashion, with an increasing number of attributes, they encounter an exponential explosion in the number of rules required for accurate prediction. This phenomenon is often referred to as the "population bloating" which hinder both efficiency and effectiveness of these approaches and poses scalability issues.

Generally, the evolutionary rule learning algorithms can be classified into three groups: Pittsburgh, Michigan, and Iterative. The Pittsburgh-style algorithms evolve a population of candidate rules, the best individual found during the evolutionary process is used to predict the class of unknown examples. Pittsburgh algorithms are traditionally used in an off-line mode where all the training data are always available a priori. Thus, real-valued attributes are normally transformed into nominal values in a pre-processing discretization step. Genetic Algorithms based claSSIfier system (GAssist) [13] is a Pittsburgh genetic-based machine learning system. The system evolves individuals that represent complete problem solutions. An individual consists of an ordered, variable-length rule set. The GAssist algorithm initialises the population at random. Real-valued attributes are represented by an adaptive discretization intervals evolved by the system simultaneously during the rule learning. This representation consists, in general terms, of a hierarchical uniform-width discretization of each attribute having different cut-points for the same attribute in different rules.

The Incremental Learning with Genetic Algorithms (ILGA) [14] is an incremental Pittsburgh rule learning algorithm. The initial population is created in either two ways: 1) the best one-condition rule set, or 2) the whole population of chromosomes in the previous solution. ILGA solves real-

valued problem on an attribute-based incremental manner such that the algorithm performs a one-dimensional search along the first attribute, trying to find the boundary information for all classes. The second step inherits this boundary information, and continues searching the boundary information for the two attributes and so on.

Michigan-style algorithms, on the other hand, evolve a set of rules which are updated as more data becomes available in an online manner. A collection of these rules comprises the solution for the classification problem. Examples of Michigan-style algorithms include the sUpervised Classification System (UCS) [10] which is an accuracy based Michigan-style LCS. The algorithm starts with an empty population and then incrementally adds rules as new data becomes available. The algorithm encodes real-valued attributes directly into GA using interval-base representation. The incremental Genetic Algorithm for mining classification rules in the presence of concept drift (IGA) [15] is also a Michigan-style evolutionary rule learning algorithm. It employs a window scheme such that training instances are accumulated until the window is full, and a genetic algorithm (GA) is then applied to determine the set of classification rules. As new training instances arrive in the window, old instances are forgotten. Once all the original instances have been replaced by new ones, the GA is re-executed to determine the new set of best classification rules. The initial population pool of the GA applied at each stage of the incremental learning process comprises a mix of the best solutions obtained in the previous stage and an appropriate number of random individuals. The algorithm only deals with nominal values and real-valued attributes are assumed to be discretised before running the algorithm.

Some algorithms have been proposed that combined both the Michigan-style and Pittsburgh-style such as COevolutionary Rule Extractor (CORE) [16]. The CORE co-evolves rules and rule sets concurrently in two cooperative populations to limit the search space and to produce good rule sets that are comprehensive. The main population encodes a population of rules using Michigan style while the co-populations are then presented with these good candidate rules to form rule sets using Pittsburgh style. The algorithm encodes real-valued attributes directly into the population using ranges and employed six comparison operators to evolve best rule sets.

Iterative rule learning algorithms create an ordered list of rules. The system iteratively applies an EA and the best individual returned is added to the end of the list of rules and all the matching instances are removed from the training data set. This process is repeated until the training data set becomes empty. HIerarchical DEcision Rules (HIDER) [17] is one example of such algorithms. It initialises the population by randomly selecting some training instances and creating rules that cover these instances. Then, the evolutionary search is run for a certain number of generations. The best rule obtained is added to the final rule set and the instances covered by the rule are removed from the training data set and the process is repeated. The algorithm encodes real-valued attributes directly into GA using the natural coding representation proposed by the author, which is a tabular representation of the computed cut-points of a specific discretization method.

III. ERL-AID DESCRIPTION

Our proposed model consists of two components running in parallel and cooperating together to find the optimal rules describing target concepts. The first component is a discretization algorithm responsible for constructing attribute intervals while the second component is a genetic algorithm searching at the rule level to find optimal combinations of intervals constructed by the first component. The discretization algorithm is used to shrink and enlarge the search space as needed online by merging/splitting attribute intervals. It is possible that some attributes will be discretized into one interval only. Thus, they are ignored by GA focusing the search on important attributes only.

While the discretization algorithm process training instances incrementally one by one, GA employs a fixed window size W, i.e. it can hold a total of W training instances. Once the window is full, GA is applied to generate new candidate rules based upon this particular set of training instances. Initial population is generated incrementally as data arrives. Once a new instance arrives, statistics of attribute intervals are updated and the population is scanned to find covering rules for the new instance. A new rule is created and inserted if there is no cover in the current population. Each time that a new training instance arrives, an old training instance is forgotten. The population of evolved rules [P] represents the solution for the classification problem in hand. The overall flow of ERL-AID is shown in Figure 1.

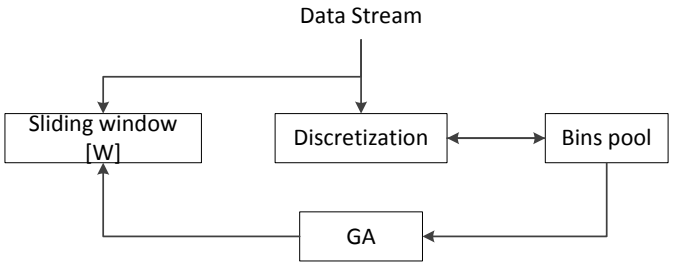


Fig. 1. High level overview of ERL-AID.

A. Online Discretization Component

A supervised incremental approach is proposed such that given a problem of *D* dimensions; each attribute a_d of the problem is initially divided into a number *B* of "micro bins" of equal sizes, where *B* is chosen to be higher number of intervals than what is usually required. A micro bin array of length *B* is maintained over time where each micro bin is a structure with four components: 1) the lower bound value, 2) the upper bound value, 3) an array of length *K* (the number of classes) to store each class frequency statistics, and 4) the macro bin index (used for mapping between original and merged bins as will be explained subsequently).

Given a new instance x_t with associated class label S^k, for each attribute value $x_t[d]$ of x_t the algorithm finds the matched micro bin (b_{micro}) and updates its class frequencies. A merge condition is checked. If the bin meets this merge condition, then the left and right adjacent bins are checked for merging.

Once two adjacent bins are qualified for merging, they form a larger bin called "macro bin". Each macro bin is a structure with three components: 1) the lower bound value, 2) the upper bound value, 3) an array of length *K* (the number of classes) to store each class frequency statistics. The newly formed macro bin is stored in a different array (macro bin array), the statistics of the newly created macro bin is then initialized from the two merged bins.

If the matched micro bin b_{micro} is already a part of a macro bin, then a split condition is checked to see weather the macroBin needs to be split. If the condition is met, the lower and upper bound values of b_{micro} are considered candidate cut points. The left and right hand side areas of the corresponding macro bin are compared and if they are statistically different, then the macro bin is partitioned into two smaller macro/micro bins.

The merge/split condition is achieved using χ^2 test which is a statistical measure used to test the hypothesis of independence of two variables. Applied to the discretization problem, it tests the hypothesis that the class attribute is independent of which of two adjacent intervals an instance belongs as follows:

$$\chi^2 = \sum_{i=1}^{m} \sum_{j=1}^{K-1} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \quad (1)$$

Where:

- *m* : is the number of bins being compared
- *K* : is the number of classes and $K-1$ is the degree of freedom
- A_{ij} : is the number of instances in the *i*th bin and *j*th class
- E_{ij} : is the expected frequency of A_{ij} and calculated as follows: $E_{ij} = \frac{R_i . S^j}{N}$ Where:
  R_i is the number of instances in *i*th interval
  S^j is the number of instances in *j*th class
  N is the total number of instances

Once we calculated χ^2, we can conclude weather the class attribute is independent of both bins or not given the level of significance chosen (i.e. $\alpha = 0.05$ or 0.1). If the conclusion of the χ^2 test is that the class is independent of the intervals, then the intervals should be merged. On the other hand, if the χ^2 test concludes that they are not independent, it indicates that the difference in relative class frequencies is statistically significant and therefore the intervals should remain separate or be separated if they are already merged.

1) Computational Requirements: In terms of memory requirements, the memory required to store micro bin array plus that required to store macro bin array is as follows:

1) micro bin array memory = number of attributes D × number of micro bins (B_{micro}) × number of classes (K)
2) macro bin array memory = number of attributes (D)× number of macro bins (B_{macro}) × number of classes (K)

Though the number of macro bins is not known a priori, in the worst case where each pair of micro bins are merged together, the size of macro bin array is $B_{macro} = B_{micro}/2$. While the best case is that there is no merge and macro bin is empty. The total memory cost is on average:

$$\begin{aligned} MC &= DK\,(B_{micro} + B_{micro}/4) \\ &= \tfrac{5}{4} DK B_{micro} \end{aligned}$$

Thus, the memory required is linearly related to the number of dimensions of the problem.

B. Genetic-based Rule Learning Algorithm

A steady state genetic algorithm is adopted to search for optimal rules. The steady state GA [18] is different from traditional genetic algorithms in that in each generation of the algorithm only one new individual is obtained and inserted into the current population and the worst individual is removed. Therefore the computational time is much smaller than the traditional one. Once the window of instances is full, GA is applied I times on the rules currently exist in $[P]$.

1) Encoding Individuals: The system evolves a population [P] of rules, each denoted R_n where $n \in 1, 2, ..., N$, and N must not exceed the maximum population size $Nmax$ set by the user. Each rule consists of a condition part, an action part (the label of the predicted class), and a set of associated parameters estimating the quality of the rule. A typical rule R_n has the form:

$$R_n = (C_n, acc_n, simp_n)$$

Where C_n is the condition part; it is a conjunction of a set of predicates. acc_n is the classification accuracy calculated as the proportion of the number of instances matched by the condition and action of the rule to its support. $simp_n$ is the simplicity of the rule calculated as the proportion of attributes removed (with flag = 0) among all attributes. A chromosome in GA represents a single rule with $D+1$ genes where each gene corresponds to one attribute and consists of two parts: 1) the index of a bin in the micro bin array, and 2) a binary flag indicates whether this attribute is being used. The last gene corresponds to the class label that this rule advocates. Figure 2 shows the general chromosome structure.

A_1	A_2	A_3	...	A_d	A_D	c_s
$Bidx_1$	$Bidx_2$	$Bidx_3$	...	$Bidx_d$	$Bidx_D$	
$flag$	$flag$	$flag$	...	$flag$	$flag$	

Fig. 2. Encoding chromosome

C. Population Initialization

Initially, the population is empty and incrementally filled with rules through a so-called "covering operator". For each new training instance, if there is no matching rule with the same class label in the current population, a new rule is created to cover this instance and inserted into the population. A covering rule is created by finding the matching micro bin for each attribute value and the class label is set to the class label of the current training instance.

1) Evaluating Individuals: The fitness of the rules in the population is calculated based on a function containing two terms namely: classification accuracy acc and simplicity. The accuracy of a rule R_n is calculated as

$$acc_n = \frac{TP}{TP + FP} \tag{2}$$

Where TP is the true positive rate, that is the number of instances correctly matched by the rule, and FP is the false positive rate, that is the number of instances incorrectly matched by the rule.

The standard way of measuring simplicity is to count the number of conditions in the rule. If a rule has at most D conditions, the simplicity of the rule (or individual) R_n can be defined as:

$$simp_n = \frac{D - u}{D} \tag{3}$$

Where u is the number of attributes that take part in the rule.

The fitness function is computed as the arithmetic weighted mean of classification accuracy and simplicity:

$$Fitness_n = w_1 * acc_n + w_2 * simp_n \tag{4}$$

Where w_1 and w_2 are weights assigned by the user. For simplicity, in this paper we assume equal weights for both components: $w_1 = 0.5$ and $w_2 = 0.5$

2) Genetic Operators: One point crossover is applied on the condition flags with a probability P_χ to construct two new offspring. Once the cut points are identified, the flags are swaped between the two parents while the actual attribute values are transferred without modification to create new two offsprings. Two mutation operators are applied on each created offspring after crossover is complete. The first mutation operator is applied with probability P_{mf} and randomly flips the flag value such that if a flag value is "0" it becomes "1" and vice versa. However, if an attribute is reported as irrelevant by the discretization algorithm (i.e. it is discretized into one interval), its flag value is always "0". Another mutation operator is applied with a probability P_{mv} to change the value of an attribute to another value, that is, change the micro bin index to another one from the micro bin index array maintained by the discretization module.

IV. Experimental Study

The focus of this analysis is to study the classification accuracy and interpretability of the proposed system, therefore the evaluation metrics used in this analysis includes the system classification accuracy and rule compactness in terms of the number of evolved rules, and the number of attributes used. In order to analyze the algorithm performance, we have carried out an experimental study on a set of data sets with varying dimensionality obtained from the UCI repository. The system performance is compared with three other evolutionary rule-based learning algorithms. Namely: sUpervised Classifier System (UCS) [10], Hierarchical Decision Rules (HIDER) [19], coevolutionary algorithm for rules discovery in data mining (CORE) [16].

A. Data Sets

The data sets used to evaluate the proposed algorithm are obtained from UCI (University of Califrnia at Irvine) Machine Learning Repositorty.

The main characteristics of the data sets are summarized in Tables I. The columns describe: the identifier of the data set (id.), the name of the data set (data set), the number of instances (#inst), the total number of features (#Att), the number of real features (#Real), the number of integer features (#int), the number of nominal features (#Nom), the number of classes (#Clas), the proportion of instances of the minority class (%Min), the proportion of instances of the majority class (%Maj).

B. Experimental Setup

The compared algorithms in this study learn incrementally from the training instances, in particular, our proposed algorithm scans the training set only once. After the training phase is finished, the algorithms are exposed to the testing set to record the classification accuracy rates. 10-fold cross validation procedure is applied to 15 data sets. Each data set is partitioned into 10 data subsets. Each time a different partition is used as the test set and the remaining nine are used as the training set.

The statistics on the training data set and test data set of the 10 runs are averaged and reported.

To analyse the statistical significance of results, a non-parametric Wilcoxon test is utilized as suggested in [20].

The parameters of the proposed algorithm used in this experimental study is as follows: The maximum population size $N = 300$, the algorithms passes one time over the training instances, the number of microbins $B = 20$, window size $W = 50$, chi-square significance threshold $\alpha = 0.05$, one-point cross over is used with crossover probability $P_\chi = 0.5$, flip mutation probability $P_{mf} = 0.2$, value mutation probability $P_{mv} = 0.1$, tournament selection is used for parent selection, number of GA runs $I = 100$.

C. Results and Comparison on Real Data Sets

This section presents the results obtained in the empirical study using 10-fold cross validation scheme. Our proposed approach ERL-AID is compared with UCS, Hider, and CORE in terms of classification accuracy, number of rules generated, number of attributes per rule, and number of conditions per rule.

a) Classification Accuracy: We compared the algorithms under investigation in terms of classification accuracy obtained on the testing data sets. The average accuracy rates are shown in Table II in which the best results are shown in bold. Acc refers to the average classification accuracy while Std indicates the standard deviation obtained. In terms of test accuracy, the results indicate that ERL-AID obtained better or comparative accuracy when compared with other methods. In particular, it got better classification results in eight out of fifteen datasets under investigations and second best in other four data sets with very slight differences. In the remaining data sets, ERL-AID also delivers competitive performance.

TABLE I
DATA SETS CHARACTERISTICS.

Id.	Name	#inst	#Att	#Real	#Int	#Nom	#clas	%Min	%Maj
mag	magic	19020	10	10	0	0	2	35.16	64.84
win	Wine	178	13	12	1	0	3	26.97	39.89
seg	segmentation	2310	19	19	0	0	7	10.65	11.43
hep	Hepatitis	155	19	2	4	13	2	20.65	79.36
rng	Ringnorm	7400	20	20	0	0	2	49.50	50.50
tnrm	Twonorm	7400	20	20	0	0	2	49.96	50.04
mus	Mushrooms	8124	20	0	0	20	2	47.67	46.68
par	Parkinson	195	22	22	0	0	2	24.62	75.38
wdbc	W. diagnose Breast cancer	569	30	30	0	0	2	37.30	62.70
ion	Ionosphere	351	34	34	0	0	2	36.90	64.10
soy	soybean-large	682	35	0	0	35	19	0.12	13.47
kr	King-Rook vs. King-Pawn	3196	36	0	0	36	2	47.78	52.22
spt	Spectf	267	44	0	44	0	2	20.60	79.40
spa	Spambase	4597	57	55	2	0	2	39.44	60.66
son	Sonar	208	60	60	0	0	2	46.67	53.33

TABLE II
CLASSIFICATION ACCURACY OF ERL-AID AND OTHER EVOLUTIONARY ALGORITHMS.

Id.	UCS		Hider		CORE		ERL-AID	
	Acc	Std	Acc	Std	Acc	Std	Acc	Std
mag	69.13	4.85	**76.08**	**0.74**	74.54	1.95	71.98	2.41
win	90.46	6.03	66.22	15.95	**93.30**	**5.44**	90.46	7.81
seg	96.67	0.89	76.90	4.30	31.39	9.96	**97.23**	**2.66**
hep	79.54	10.27	**84.03**	**14.04**	79.38	3.94	83.87	8.64
ring	50.42	0.31	40.00	0.00	62.93	2.60	**73.38**	**4.71**
tnrm	70.74	16.95	69.97	3.64	67.92	2.28	**79.09**	**2.14**
mus	97.12	0.64	90.00	0.94	87.58	3.98	**98.55**	**0.98**
par	79.45	7.93	**56.91**	**15.08**	75.39	1.87	83.95	9.13
wdbc	**93.33**	**4.59**	44.56	9.00	62.74	0.70	93.15	3.91
ion	86.33	5.68	78.50	4.90	63.24	5.97	**88.32**	**6.24**
soy	32.00	1.70	**85.98**	**3.85**	14.49	4.20	63.41	5.76
kr	83.64	2.02	**94.31**	**1.31**	59.14	2.72	91.46	2.51
spt	75.30	4.96	45.92	7.27	**79.42**	**1.75**	79.07	5.16
spa	**93.20**	**0.70**	94.60	1.30	60.60	0.00	59.49	3.04
son	52.40	11.20	57.50	7.60	53.38	1.71	**68.69**	**9.34**
avg	76.65	5.25	70.77	5.99	64.36	3.27	**81.47**	**4.96**

TABLE III
WILCOXON TEST OF ACCURACY (ERL-AID IS THE CONTROL ALGORITHM)

Algorithm	p-value	Hypothesis
UCS	0.01660	Rejected
HIDER	0.02155	Rejected
CORE	0.00263	Rejected

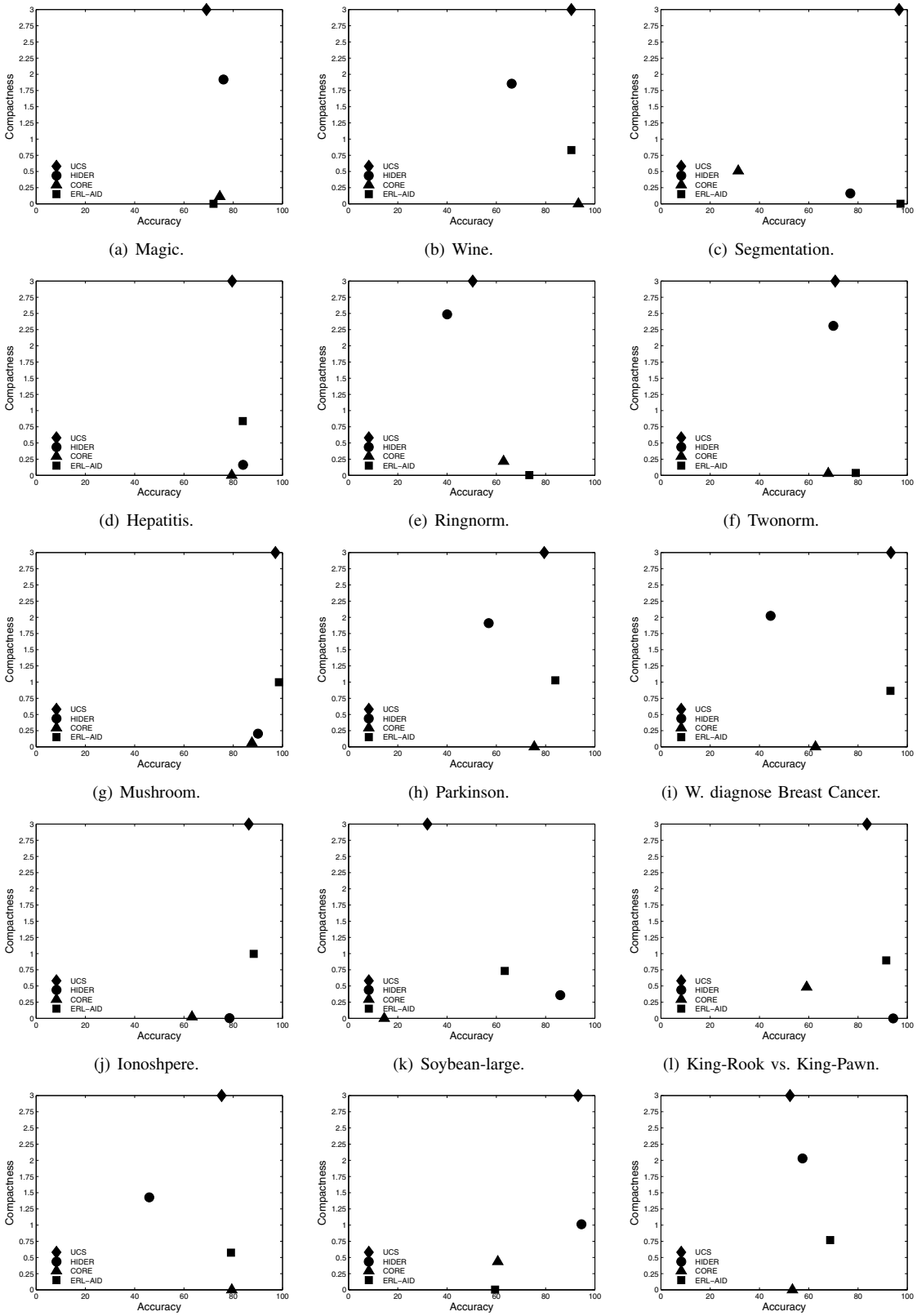
Since the obtained results may not present normal distribution or homogeneity of variance, we consider the use of non-parametric Wilcoxon signed-rank test to find significant differences among the results obtained by the proposed algorithm and the other algorithms. Table III indicates that p-value are smaller than the significance level chosen 0.05, so the hypothesis of equivalence of results is rejected with all other algorithms in the comparison. Thus, ERL-AID is considered statistically better than the other algorithms in terms of classification accuracy.

b) Rule Compactness: The compactness results are shown in Table IV, where #rules is the average number of rules, #att is the average number of attributes per rule, #cond is the average number of conditions per rule respectively. It must be pointed out that CORE algorithm learns DNF rules (CORE), while the other methods learn non-DNF rules. A DNF rule is a special type of rule which can comprise several simple rules together. This explains the reason why The number of conditions reported for the UCS, Hider, and ERL-AID algorithms is the same as the number of attributes. CI is a compactness index computed in order to measure the compactness of generated rules as suggested in [21], with the following expression:

$$CI = R_{nrmlz} + A_{nrmlz} + C_{nrmlz} \tag{5}$$

Where R_{nrmlz}; A_{nrmlz} and C_{nrmlz} are the normalized values for the number of rules, number of attributes per rule, and number of conditions per rule, respectively. The lower the value of CI the better the rule compactness. Table V showed that obtained Wilcoxon two-sided p-value are smaller than the significance level chosen 0.05. Thus the hypothesis of equivalence of rule compactness with other algorithms is rejected. CORE is statistically the best algorithm in terms of rule compactness while ERL-AID algorithm is statistically the second best algorithm in this regard. Although CORE algorithm is the best algorithm regarding compactness, its classification accuracy was the worst among all four algorithms. While both the classification accuracy and the rule compactness are desired objectives, achieving good rule compactness at the expense of the classification accuracy is not the ultimate goal. Figure 3 highlights the accuracy vs. compactness tradeoff for the four algorithms on each data set under investigation where the x-axis represents the average accuracy obtained by each algorithm and the y-axis represents the compactness index computed according to Equation 5.

Overall, the results show that the proposed system can provide a good balance between classification accuracy and interpretability while learning incrementally from real-valued data sets.



(a) Magic. (b) Wine. (c) Segmentation.

(d) Hepatitis. (e) Ringnorm. (f) Twonorm.

(g) Mushroom. (h) Parkinson. (i) W. diagnose Breast Cancer.

(j) Ionoshpere. (k) Soybean-large. (l) King-Rook vs. King-Pawn.

(m) Spectf. (n) Spambase. (o) Sonar.

Fig. 3. Accuracy Vs. Compactness for each data set

TABLE IV
RULE COMPACTNESS OF THE ERL-AID AND OTHER EVOLUTIONARY ALGORITHMS.

Id.	UCS				HIDER				CORE				ERL-AID			
	#rules	#Att	#cond	CI	#rules	#Att	#cond	CI	#rules	#Att	#cond	CI	#rules	#Att	#cond	CI
mag	5468.50	10	10	3	47.90	9.63	9.63	1.92	3.67	1.73	2.05	0.11	7.40	1.41	1.41	**0.00**
win	3376.40	13	13	3	27.70	12.36	12.36	1.86	3.00	4.50	4.56	**0.00**	81.40	7.95	7.95	0.83
seg	6074.70	19	19	3	4.91	3.00	3.00	0.16	3.17	5.99	6.06	0.51	11.30	1.60	1.60	**0.00**
hep	2599.20	19	19	3	4.33	6.50	6.50	0.16	6.33	4.77	5.99	**0.00**	72.90	10.93	10.93	0.84
ring	5863.60	20	20	3	3202.00	19.53	19.53	2.49	10.50	4.71	6.64	0.22	29.30	3.94	3.94	**0.00**
tnrm	5899.30	20	20	3	2568.00	19.00	19.00	2.31	7.50	4.05	5.03	**0.03**	27.50	4.59	4.59	0.04
mus	6211.20	22	22	3	6.23	7.10	7.10	0.20	6.00	3.32	8.00	**0.06**	205.60	13.43	13.43	1.00
par	4306.40	22	22	3	44.40	21.00	21.00	1.91	1.00	2.00	2.00	**0.00**	76.40	12.10	12.10	1.03
wdbc	4301.70	30	30	3	180.70	29.75	29.75	2.02	1.00	5.00	5.00	**0.00**	83.80	15.56	15.56	0.86
ion	4586.10	34	34	3	8.90	5.80	5.80	**0.00**	1.67	5.80	6.43	0.02	141.50	19.40	19.40	1.00
soy	5486.5	35	35	3	20.60	8.67	8.67	0.36	1.00	3.00	3.00	**0.00**	198.40	14.13	14.13	0.73
kr	6070.10	36	36	3	3.00	2.40	2.40	**0.00**	14.00	8.86	12.07	0.48	240.80	16.77	16.77	0.89
spt	5494.00	44	44	3	70.00	36.09	36.09	1.43	1.00	17.00	17.00	**0.00**	108.40	24.49	24.49	0.57
spa	6300.90	57	57	3	15.60	31.65	31.65	1.01	1.00	17.00	17.00	0.44	21.30	5.85	5.85	**0.00**
son	5928.60	60	60	3	178.20	59.97	59.97	2.03	1.00	17.50	17.50	**0.00**	137.00	33.33	33.33	0.77
avg				3				1.19				0.12				0.57

TABLE V
WILCOXON TEST OF COMPACTNESS (ERL-AID IS THE CONTROL ALGORITHM)

Algorithm	p-value	Hypothesis
UCS	0.00	Rejected
HIDER	0.04126	Rejected
CORE	0.01025	Rejected

V. CONCLUSIONS AND FUTURE WORK

We presented a new incremental algorithm based on Chi-square statistical test and GA for solving real-valued classification problems. The algorithm performs attribute discretization incrementally and in parallel with evolutionary rule searching which also includes an embedded feature selection.

An experimental study involving three well-known evolutionary learning algorithms has been carried out on 15 data sets, and classification accuracy and compactness have been compared and analyzed. Comparative results show that the proposed method produces better or comparable classification accuracy with concise set of rules for the data sets being tested. As future work, it would be of interest to perform an extensive study on extremely high dimensional problems. It is also important to compare different discretization approaches and study the effects on the overall system performance. The use of Multi-objective evolutionary algorithms are also possible direction for future research since the current problem is comprised of two goals.

REFERENCES

[1] A. A. Freitas, "A genetic programming framework for two data mining tasks: classification and generalized rule induction," in *Genetic Programming 1997: Proc 2nd Annual Conf.* Morgan Kaufmann, 1997, pp. 96–101.

[2] ——, "A review of evolutionary algorithms for data mining," in *Soft Computing for Knowledge Discovery and Data Mining*. Springer, 2008, pp. 79–111.

[3] S. W. Wilson, "Classifier fitness based on accuracy," *Evol. Comput.*, vol. 3, no. 2, pp. 149–175, 1995.

[4] W. Ruojun, C. Duwu, and Z. Ye, "Rule induction based on a novel evolutionary strategy," in *Intelligent Control and Automation, 2002. Proceedings of the 4th World Congress on*, vol. 4. IEEE, 2002, pp. 3171–3174.

[5] A. A. Freitas, *Data mining and knowledge discovery with evolutionary algorithms*. Springer, 2002.

[6] K. A. DeJong and W. M. Spears, "Learning concept classification rules using genetic algorithms," DTIC Document, Tech. Rep., 1990.

[7] K. Shafi, T. Kovacs, H. Abbass, and W. Zhu, "Intrusion detection with evolutionary learning classifier systems," *Natural Computing*, vol. 8, no. 1, pp. 3–27, 2009.

[8] E. Debie, K. Shafi, C. Lokan, and K. Merrick, "Performance analysis of rough set ensemble of learning classifier systems with differential evolution based rule discovery," *Evolutionary Intelligence*, pp. 1–18, 2013.

[9] J. H. Holland, "Adaptation," in *Progress in Theoretical Biology IV*, A. Press, Ed. Academic, New York, 1976, pp. 263–293.

[10] E. Bernado, Mansilla, and J. M. Garrell-Guiu, "Accuracy-based learning classifier systems: models, analysis and applications to classification tasks," *Evol. Comput.*, vol. 11, no. 3, pp. 209–238, 2003.

[11] C. Zhou, W. Xiao, T. M. Tirpak, and P. C. Nelson, "Evolving accurate and compact classification rules with gene expression programming," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 6, pp. 519–531, 2003.

[12] S. Kotsiantis and D. Kanellopoulos, "Discretization techniques: A recent survey," *GESTS International Transactions on Computer Science and Engineering*, vol. 32, no. 1, pp. 47–58, 2006.

[13] J. Bacardit and J. M. Garrell, "Evolving multiple discretizations with adaptive intervals for a pittsburgh rule-based learning classifier system," in *Genetic and Evolutionary ComputationGECCO 2003*. Springer, 2003, pp. 1818–1831.

[14] S.-U. Guan and F. Zhu, "An incremental approach to genetic-algorithms-based classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 2, pp. 227–239, 2005.

[15] I.-H. Li, I.-E. Liao, and W.-Z. Pang, "Mining classification rules in the presence of concept drift with an incremental genetic algorithm," *Journal of Theoretical & Applied Information Technology*, vol. 4, no. 7, 2008.

[16] K. Tan, Q. Yu, and J. Ang, "A coevolutionary algorithm for rules discovery in data mining," *International Journal of Systems Science*, vol. 37, no. 12, pp. 835–864, 2006.

[17] J. Aguilar-Ruiz, J. Riquelme, and M. Toro, "Evolutionary learning of hierarchical decision rules." *Transactions on Systems and Man and and Cybernetics - Part B: Cybernetics*, vol. 33, no. 2, pp. 324–331, 2003.

[18] D. Whitley and J. Kauth, *GENITOR: A different genetic algorithm*. Colorado State University, Department of Computer Science, 1988.

[19] J. Aguilar-Ruiz, R. Giráldez, and J. Riquelme, "Natural encoding for evolutionary supervised learning," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 4, pp. 466–479, 2007.

[20] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1248547.1248548

[21] F. J. Berlanga, A. Rivera, M. J. del Jesús, and F. Herrera, "Gp-coach: Genetic programming-based learning of compact and accurate fuzzy rule-based classification systems for high-dimensional problems," *Information Sciences*, vol. 180, no. 8, pp. 1183–1200, 2010.