2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Towards better generalization in Pittsburgh Learning Classifier Systems

Shubhra Kanti Karmaker Santu, Md. Mustafizur Rahman, Md. Monirul Islam and Kazuyuki Murase

***Abstract*—Generalization ability of a classifier is an important issue for any classification task. This paper proposes a new evolutionary system, i.e., EDARIC, based on the Pittsburgh approach for evolutionary machine learning and classification. The new system uses a destructive approach that starts with large-sized rules and gradually decreases the sizes as evolution progresses. Unlike most previous works, EDARIC adopts an intelligent deletion mechanism, evolves a separate population for each class of a given problem and uses an ensemble system to classify unknown instances. These features help in avoiding over-fitting and class-imbalance problems, which are beneficial for improving generalization ability of a classification system. EDARIC also applies a rule post-processing step to exempt the evolution phase from the burden of tuning a large number of parameters. Experimental results on various benchmark classification problems reveal that EDARIC has better generalization ability in case of both standard and imbalanced datasets compared to many existing algorithms in the literature.**

I. Introduction

Generalization ability of a classifier is an important issue for any classification task. It determines accuracy of the classification system on unknown instances. Two prominent problems affecting the generalization ability are over-fitting and class-imbalance. It is thus important to address these problems while developing the classification system.

Evolutionary rule-based classification systems, also called Learning Classifier Systems (LCSs), are promising methods for rule extraction and classification. Unlike neural networks and support vector machines, LCSs are suitable for many practical applications (e.g. churn prediction) due to their ability in producing human understandable *if-then* rules for classification. The application of evolutionary algorithms to classification domains has traditionally been addressed from two different points of view: the Pittsburgh approach (or Pittsburgh LCS) and the Michigan approach (or Michigan LCS). We are particularly interested in the Pittsburgh approach for this work.

The Pittsburgh approach usually evolves a population containing individuals where each individual consists of a set of variable-length classification rules and forms a candidate solution to a given classification task. Some popular Pittsburgh approaches are DMEL [2], GAssist [3], [4], pitts-GIRLA [12], ILGA [18], GIL [19], and OIGA [30]. Some of them (e.g. [4], [12] and [19]) explicitly deal with generalization, while others (e.g. [2], [18] and [30]) do not. Bacardit *et. al.* [4] used the minimum description length principle for generalization pressure and bloat control. Pitts-GIRLA [12] and GIL [19] employed a random deletion mechanism for better generalization. However, random deletion may remove important rules from a rule-set (or constraints from a rule), which may make evolution inefficient. Furthermore, a comprehensive attempt to deal with the class-imbalance problem is still missing in case of Pittsburgh approaches, although Michigan approaches have been adapted to address this problem [22], [23].

In this work, we present a new evolutionary system, called *Evolutionary Destructive Approach to Rule Induction and Classification* (EDARIC), based on the Pittsburgh approach. Our new system is different from most existing Pittsburgh LCSs on a number of aspects. First, it not only addresses the over-fitting problem but also the class-imbalance problem. Second, it uses a destructive approach with an intelligent deletion mechanism for producing general (instead of specific) rule-sets with an aim of avoiding over-fitting. Third, EDARIC evolves a separate population for each class of a given problem and finally uses an ensemble system for handling the class-imbalance problem. Fourth, EDARIC applies a rule post-processing step that exempt the evolution phase from the burden of tuning a large number of parameters which requires running an evolutionary system multiple times with different parameter settings. Fifth, EDARIC has the beauty of simplicity that uses few simple operators and parameters.

This is how the rest of the paper is organized. Section II describes the algorithm EDARIC in details with suitable examples for each operator used there. Section III reveals our experimental setup and results. Section IV draws our conclusion and possible future studies.

II. The EDARIC algorithm

The destructive process of our algorithm starts evolution from the most specific rule-sets and then gradually makes them as much generic as possible by deleting their less interesting constraints (attributes). The deletion incorporated here works on the entire population, not on a specific rule-set or rule. This kind of batch deletion is a unique feature of this work.

EDARIC evolves a separate population, P_i, for a particular class i of a given classification problem. The rationale behind maintaining multiple populations is that, accuracy based classifiers are often biased towards the major class, neglecting significant information about the minor classes, especially

Shubhra Kanti Karmaker Santu, Md. Mustafizur Rahman and Md. Monirul Islam are with the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh. (email:{kantishubhra, mustafiz_rahman, mdmonirulislam}@cse.buet.ac.bd).

Kazuyuki Murase is with the Department of Human and Artificial Intelligence Systems, University of Fukui, Bunkyo, Fukui, Japan. (email: murase@u-fukui.ac.jp).

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

for the class-imbalanced problems. By maintaining multiple populations, our algorithm gives equal importance to all the classes, thus preserves information about both the major and minor classes. Different components of the proposed algorithm are described below.

A. Initialization and Encoding

Each population, P_i, contains ψ individuals, each of which is a rule-set consisting of several rules. Our EDARIC initializes the rule-set by randomly copying $m\%$ of the training instances having the class label i. The parameters m and ψ are defined by the user. The advantages of this initialization process are that it is very simple and does not cost any computation overhead. Most importantly, by such an initialization, start of evolution with the most specific rules is ensured.

EDARIC follows the Pittsburgh approach for encoding and evolves a separate P_i for each class i of a given problem. Each rule-set of P_i represents a candidate solution to the class i. A rule simply encodes a conjunction of constraints on the attributes of a training instance. We use #, the wild-card character, to represent a deleted constraint. EDARIC assumes discrete data. So, continuous data is discretized before the application of EDARIC.

B. Evaluation mechanism

A fitness function, F_{Ch}, is used to evaluate each individual, Ch, of P_i. As EDARIC adopts the destructive approach, this function must incorporate the accuracy of Ch as well as the overall amount of constraints (attributes) deleted from it. Under this condition, F_{Ch} must be a monotonically increasing function of both accuracy and deletion. Thus it can be defined as

$$F_{Ch} = \alpha * A_{Ch} + (1 - \alpha) * D_{Ch} \tag{1}$$

where A_{Ch} is the percentage accuracy of Ch, D_{Ch} is the percentage of constraints deleted from Ch and α is a weight. The user-defined parameter α lies in $0 \leq \alpha \leq 1$ and determines the impact of accuracy and deletion on F_{Ch}. For a particular accuracy and α, it is obvious that the higher the amount of deletion is, the greater the fitness is. Thus, assuming that the accuracy of two individuals (rule-sets) is same, the fitness of a general rule-set that contains a fewer number of rules and/or constraints will always be higher than that of a specific one.

C. Evolutionary operators

1) Selection: The selection operation is performed using the traditional roulette wheel selection, also known as fitness proportionate selection [15], to choose individuals for crossover and mutation operations. The individuals of a population are selected for genetic operations with a probability proportional to their fitness value defined by Eq. (1). That is, the individuals that are more fit have a greater probability of being selected.

2) Crossover: EDARIC uses the two-point crossover for breeding. Besides interchanging constraints (attributes) between two individuals, crossover allows re-occurrence of deleted constraints (attributes) with possibly the same or different values. Figure 1 demonstrates this process. Notice that, the second rule of the first individual has two attributes and one attribute, respectively, before and after the application of crossover. And the deleted attribute in the first rule of the first individual appears again after crossover and gets the value y_4.

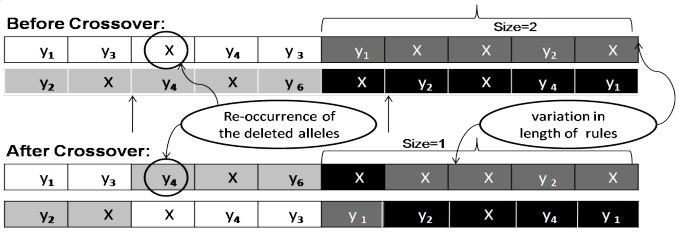


Fig. 1: Two-point Crossover operator of EDARIC

3) Mutation: The mutation operator randomly selects a constraint (attribute) from a randomly chosen rule and changes the constraint randomly to an allowable one with a small mutation probability. Besides this traditional role, it tries to make the rule more generic, performs a biased deletion with a bias towards less interesting attributes. To implement this, we define interestingness of an attribute and formulate a selection mechanism for deleting the less interesting attributes.

We first consider the interestingness of a particular value of an attribute and then use it to derive the overall interestingness of the attribute. For a particular value l of the attribute atr and the class label r, the interestingness of a rule *if atr = l then class = r* can be defined by the RI (Rule Interestingness) measure as:

$$RI_{atr,l,r} = N_{atr,l,r} - N_{atr,l} \times \frac{N_r}{N} \tag{2}$$

where, N_r= Number of examples with class value r
$N_{atr,l}$= Number of examples with value l for attribute atr
$N_{atr,l,r}$= Number of examples with class value r and value l for attribute atr
N=Total number of training examples

The insight behind Eq.(2) can be gained by considering the fact that $\frac{N_r}{N}$ is the number of times in percentage class value r occurs in the training data. So, $N_{atr,l} \times \frac{N_r}{N}$ denotes the expected number of times l and r appear together in a training example. Whereas, $N_{atr,l,r}$ is the actual number of times l and r appear together. So, Eq.(2) measures rule interestingness as a difference between the observed and the expected value.

We are now ready to define the overall interestingness of an attribute by considering all the possible left and right hand combinations. The RI of atr can be defined as

$$RI_{atr} = \frac{\sum_l \sum_r RI_{atr,l,r}}{n \times \kappa} \tag{3}$$

Here, κ is the cardinality of the set for all possible values (discrete) of atr and n is the total number of classes. Finally, we normalize RI_{atr} by dividing it N.

Algorithm 1 Evolutionary Destructive Approach to Rule Induction and Classification

Calculate RI_j for each attribute j
for $i = 1 \rightarrow Total\_Number\_Of\_Classes$ **do**
  $P_i \leftarrow$ Initial Population for class i
  $\Delta_0 \leftarrow$ initial deletion amount for P_i
  set $RW_{j,0}$ to $1/RI_j$ for each attribute j
  $q \leftarrow 0$;
  while Better solution can be found **do**
    Delete Δ_q alleles of P_i using intelligent deletion;
    Execute GA to optimize P_i
    $\Delta_{q+1} \leftarrow$ revised deletion amount for next iteration;
    $RW_{j,q+1} \leftarrow$ deletion probability in next iteration for each attribute j;
    $q \leftarrow q + 1$;
  end while
  Add the rules from $Ch(best, q)$ with $RI > \theta_i$ to final rule-set for class i
end for

$$RI_{atr} \leftarrow \frac{RI_{atr}}{N} \tag{4}$$

Now, let's define the number of attributes that are to be deleted from population P_i in the initial iteration. We select this number to be a portion (preferably small) of the total number of alleles in the individual. Thus, we define the initial deletion amount Δ_0 by Eq. (5).

$$\Delta_0 \leftarrow \gamma * \left\{ \sum_{k=1}^{TG(Ch,i)} N_k(Ch, i) \right\} \tag{5}$$

Here, for the population P_i, $N_k(Ch, i)$ and $TG(Ch, i)$ denote the number of attributes in the rule k of Ch and the total number of rules in Ch respectively.

An obvious question is which attributes are to be deleted. We adopt a stochastic approach here rather than a deterministic one. A selection procedure is implemented through roulette wheel in such a way that the deletion probability of the less interesting attributes be higher than that of the more interesting attributes. This is easily implemented by initializing the roulette wheel by the $1/RI_{atr}$ measure (normalized) for the probability of each attribute to be selected for deletion. That is, the probability $RW_{atr,init}$, that attribute atr be selected for deletion at the initial stage of the rule generalization process through the roulette wheel mechanism is defined by the following equation:

$$RW_{atr,init} = 1/RI_{atr} \tag{6}$$

We thus define a biased deletion procedure with a bias towards less interesting attributes at the initial stage of the rule generalization process. We randomly select one rule of some individual and delete that particular attribute of this rule which is selected by the roulette wheel selection method as discussed above. This procedure is repeated Δ_0 times which performs the batch deletion of Δ_0 attributes over the entire population P_i.

Now, another question that immediately arises is that what will be the deletion amount and deletion criteria for the next iterations. It is obvious that, at initial stages we would prefer a larger scale deletion than at final stages where the candidate solutions have nearly converged. An adaptive method for determining the deletion rate at every iteration is needed that will depend on the initial deletion rate, the change in accuracy due to the previous deletion process and the overall deletion amount. All these points are incorporated in our Eq. (7). Here, $Ch(best, q)$ is the fittest individual after iteration q. $D_{Ch(best,q)}$ is the percentage of attributes already deleted from the fittest chromosome after iteration q, $A_{Ch(best,q)}$ is the accuracy of the fittest chromosome after iteration q and $A_{Ch(best,q-1)}$ is the accuracy of fittest chromosome after iteration $q - 1$. Eq. (7) drives Δ_{q+1} to decrease in value when the percentage of attributes already deleted increases and drives the same to increase in value when accuracy of the fittest chromosome in the next iteration increases.

$$\Delta_{q+1} \leftarrow \Delta_0 * \left\{ 1 - D_{Ch(best,q)} + A_{Ch(best,q)} - A_{Ch(best,q-1)} \right\} \tag{7}$$

We now focus on which attributes are going to be deleted. It directly indicates that the roulette wheel selection mechanism for the biased deletion must be modified in some manner that directly reflects the refined probability of each attribute to be deleted. We will ultimately move towards uniformity from biasness, but the key question is what should be the rate by which we should approach uniformity. Δ_q is obviously a guideline. Standard deviation of the attribute's interestingness may be another guideline. If Δ_q is high, we are away from uniformity and should approach uniformity at a higher rate than if this value is low. So, the probabilities in the roulette wheel selection mechanism can be iteratively modified by Eq.(8). Here, RW_U is the value of the probability that each attribute must have in a roulette wheel selection mechanism to ensure uniform selection for deletion and $RW_{atr,q}$ is the roulette wheel selection probability of attribute atr to be selected for deletion after iteration q. Eq.(8) forces $RW_{atr,q}$ to be uniform eventually for each attribute atr with a rate proportional to Δ_q. Thus, uniformity is achieved automatically after a number of iterations. Normalization of RW values is again a must.

$$RW_{atr,q+1} \leftarrow RW_{atr,q} + \Delta_q * (RW_U - RW_{atr,q}) \tag{8}$$

Our EDARIC is terminated in the following two cases. Firstly, if no better solution could be found after μ_1 iterations. Secondly, if any further deletion action incurs severe penalty in accuracy. The pseudo code of EDARIC is given in Algorithm 1. This algorithm runs in $O(n\mu_1\mu_2)$ time where n is the total number of classes, μ_1 is number of times the while loop executes and μ_2 is number of iterations GA executes each time it is evoked. The running time of Algorithm 1 is independent w.r.t. population size as steady state GA is used rather than traditional generational GA.

D. Post-processing, ensemble system and classification

The classification task comes into action after the Rule Induction task evolves the final rule-sets for all the classes. As EDARIC is based on the Pittsburgh approach and evolves a separate population P_i for each class i, the best individual (rule-set) of P_i is the solution for the class i. From this rule-set, rules with RI measure above θ_i are retained and the rest are discarded, where θ_i is a user defined parameter. The evolution session is now relieved from the burden of time consuming multiple runs with different parameter settings (specially α). This is because θ_i works on the evolved concise rule set and EDARIC can tune it by varying its value, which has almost the same effect as varying α during evolution.

Algorithm 2 EDARIC: Classification Task

$R \leftarrow \{\}$
for $i = 1 \rightarrow Total\_Number\_Of\_Classes$ **do**
  for each gene g of the best individual in P_i **do**
    Construct a rule r of the form $g \rightarrow i$
    Calculate RI of r
    if $RI > \theta_i$ **then**
      $R \leftarrow R \cup r$
    end if
  end for
end for
for each testing example k **do**
  match k with each rule $r \epsilon R$
  Let, c be the class with maximum number of counts as the consequent of the matched rules
  if c is unique **then**
    assign k to class c
  else
    assign k to class c_p, where, c_p be the class with maximum number of counts as the consequents of the partially matched rules. Resolve ties randomly.
  end if
end for

The retained rules of each P_i are listed together and are used as an ensemble system to classify unknown instances that are not used during evolution. An unknown instance is matched against all the rules and the class with the maximum number of matching rules is assigned as its class. In case of a tie, partial matches (w.r.t. number of constraints matched) are calculated and the class with the maximum partial match is selected. If observed carefully, it can easily be identified that θ_i can also work as an essential component to handle the class-imbalance problem. Consider the ensemble system that applies majority voting; keeping θ_i low, i.e., allow more rules to retain, for the minor classes and keeping the same high, i.e., allow fewer rules to retain, for the major classes makes the ensemble more conservative(flexible) in classifying an instance as a major(minor) class. Thus, θ_i add an extra layer to handle the class-imbalance problem. The overall methodology is given in Algorithm 2, which shows that classification of a particular testing sample runs in linear time w.r.t. the number of rules learnt.

TABLE I: Performance of EDARIC on different problems

Standard Datasets			Imbalanced datasets	
Dataset	**Accuracy**	**Kappa**	**Dataset**	**G-mean**
abalone	27.85±4.76	0.16±0.05	Abalone9-18	69.62 ± 8.87
australian credit	87.82±1.26	0.75±0.02	Abalone19	72.17 ± 9.57
liver disorders [Bupa]	69.52±3.49	0.36±0.07	Ecoli0vs1	98.64 ± 1.66
cleveland	60.13±3.03	0.33±0.06	Ecoli3	86.47 ± 10.6
contraceptive	55.18±1.23	0.29±0.02	Glass0	85.49 ± 4.07
japanese credit	88.49±3.00	0.76±0.06	Glass2	67.24 ± 9.62
ecoli	81.13±5.09	0.73±0.06	Glass5	94.86 ± 2.25
german credit data	75.19±1.36	0.33±0.04	Haberman	62.99 ± 4.99
glass identification	73.33±3.74	0.64±0.05	Iris0	100.0 ± 0.0
haberman	74.36±2.01	0.18±0.15	New-Thyroid1	100.0 ± 0.0
heart	85.28±5.3	0.69±0.11	Page-Blocks0	92.01 ± 0.61
hepatitis	93.86±6.23	0.69±0.37	Pima	77.99 ± 3.85
iris	95.31±3.21	0.92±0.04	Vehicle1	71.76 ± 3.93
lymphography	85.24±5.64	0.71±0.10	Vehicle3	70.21 ± 3.33
new-thyroid	97.99±2.39	0.95±0.05	Vowel0	97.62 ± 2.1
nursery	85.77±1.48	0.78±0.02	Wisconsin	97.73 ± 0.38
pima	79.13±2.57	0.51±0.06	Yeast2	69.8 ± 2.78
ring	94.99±1.46	0.89±0.02	Yeast4	80.99 ± 4.25
vehicle	73.11±2.49	0.64±0.03	Yeast5	95.93 ± 2.3
wine	98.51±2.15	0.97±0.03	Yeast6	88.34 ± 7.5

TABLE II: Percentage deletion (PD) and Compactness (Comp) results for EDARIC

Standard Datasets			Imbalanced datasets		
Dataset	**PD**	**Comp**	**Dataset**	**PD**	**Comp**
abalone	0.9483	0.8291	Abalone9-18	0.9935	0.9883
australian credit	0.8911	0.8103	Abalone19	0.9984	0.9963
liver disorders [Bupa]	0.6791	0.6724	Ecoli0vs1	0.9893	0.9760
cleveland	0.9350	0.9066	Ecoli3	0.99	0.9801
contraceptive	0.8746	0.8366	Glass0	0.9720	0.9379
japanese credit	0.8471	0.6807	Glass2	0.9755	0.9502
ecoli	0.9384	0.8887	Glass5	0.9931	0.9865
german credit data	0.8719	0.6972	Haberman	0.9749	0.9711
glass identification	0.8710	0.7798	Iris0	0.9681	0.9436
haberman	0.9917	0.9849	New-Thyroid1	0.9759	0.9522
heart	0.7617	0.5943	Page-Blocks0	0.9899	0.9747
hepatitis	0.9021	0.6707	Pima	0.9661	0.9546
iris	0.9726	0.9418	Vehicle1	0.9285	0.8305
lymphography	0.8859	0.6129	Vehicle3	0.9437	0.8403
new-thyroid	0.9503	0.9069	Vowel0	0.9880	0.9731
nursery	0.8359	0.8016	Wisconsin	0.9731	0.9373
pima	0.9746	0.9618	Yeast2	0.9939	0.9912
ring	0.9592	0.8407	Yeast4	0.9984	0.9976
vehicle	0.8543	0.6099	Yeast5	0.9980	0.9968
wine	0.9033	0.7227	Yeast6	0.9992	0.9991

TABLE III: Wilcoxon signed rank test summary between EDARIC with selective deletion and EDARIC with random deletion. R^+ corresponds to the sum of ranks for EDARIC with selective deletion and R^- for EDARIC with random deletion.

test	R^+	R^-	**Hypothesis**($\alpha = 0.05$)	**p-value**
accuracy	28	0	Rejected for selective deletion	0.017960
kappa	28	0	Rejected for selective deletion	0.017960
g-mean	41	4	Rejected for selective deletion	0.028402

III. Experimental Studies

In this section, we evaluate and compare the performance of EDARIC on 20 standard and 20 imbalanced classification problems. These problems have been the subject of many studies in the machine learning society. The detailed description of these problems can be obtained from the University of California Irvine Machine Learning Repository, *http://archive.ics.uci.edu/ml/datasets.html*.

We followed the guidelines provided by Fernandez *et al.* (2010) [14] to conduct experiments and used the same exact same cross-validation partitions as was used there for a fair comparison. We converted the continuous attributes into discrete ones for EDARIC by using the Fayyad and Irani's MDL discretization scheme [5], which was implemented through a data processing tool named *Orange* [13]. To compare EDARIC's performance with the other algorithms, we used the accuracy and Cohen's kappa measure for the standard (balanced) data-sets and g-mean for imbalanced datasets. For each data-set, we applied 5-iterated 5-fold cross validation and considered the average results of 25 runs for comparison with other works reported in [14]. To demonstrate the generalization power of EDARIC, We also define some other performance measures, like:

Compactness: The term *compactness* signifies how much compact the learned rule-set is. If the size of the learned rule-set is $x\%$ of the training dataset, then the rule-set is $(100-x)\%$ compact. Mathematically,

$$Compactness = 1 - \frac{N_r}{N} \tag{9}$$

Here, N_r is number of unique rules learnt and N is Number of training rows.

Percentage deletion: The term *percentage deletion* (*PD*) stands for the portion of alleles deleted from the best *individual* to produce the unique generalized rules. Mathematically,

$$PD = 1 - \frac{NA_r}{NA_c} \tag{10}$$

Here, NA_r is total number of conjuncts in the final rule-set and NA_c is total number of alleles in any individual at the initial stage of evolution.

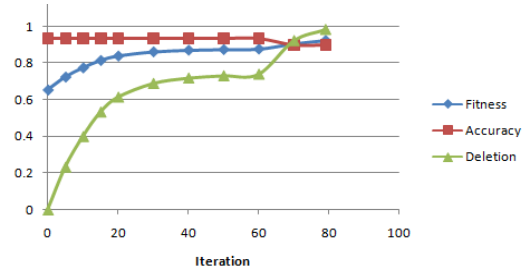
As mentioned earlier, EDARIC uses very few parameters. Important training parameters are α, γ and m which were set to the values 0.7 (this means to put more weight on classifier accuracy for fitness evaluation), 0.02 (a small constant) and 40 (midway between over-sampling and under-sampling) respectively. We did not change the values of any of these parameters during evolution. However, θ was varied between 0 to 0.2 in the post-processing step.

A. Results

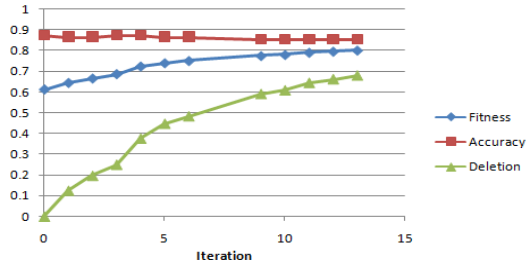
Table I shows the average performance of EDARIC on different standard and imbalanced classification problems. It is evident that EDARIC has a good testing accuracy, kappa and g-mean. For example, in case of wine data-set, the average testing accuracy of rules evolved by our method was 98.51%, while it was 79.13% for the pima data-set.

Table II shows the Percentage Deletion (PD) and Compactness (Comp) measures obtained by EDARIC for each data-set that clearly demonstrates the generalization capabilities of EDARIC. For example, for the Ecoli3 dataset, the evolved rule-set was 98.01% compact and 99% of the alleles of the best individual were deleted to create the general rules.

We now try to provide an experimental evidence of how the algorithm converges in Fig. 2 by demonstrating the evolution



(a) negative class



(b) positive class

Fig. 2: Convergence for ecoli3 dataset

of separate populations for a particular dataset, Ecoli3. Clearly evident from Fig.(2), as the number of generations increases, the deletion amount as well as fitness value increases for both positive and negative class. Note that, our algorithm sometimes sacrifices higher training accuracy in order to keep the evolved rule-set as general as possible (Fig. 2(a)).

B. Effect of selected deletion

To study the effects of selected deletion, we experimented with a variation of EDARIC where the deletion of attributes was done randomly instead of using the intelligent deletion mechanism proposed in this work. All other components of the two variants were same. For space constraints, we do not present results for all the data-sets; rather we present results for seven standard and nine imbalanced data-sets (Table IV).

From Tables IV, it can be observed that EDARIC with selective deletion has better performance than EDARIC with random deletion almost all the times. To see whether this difference is statistically significant, we applied wilcoxon-signed-rank-test on these results. Table III shows the summary of the statistical test. Here, R^+ corresponds to the sum of ranks for EDARIC with selective deletion and R^- for EDARIC with random deletion. It shows that the null hypothesis has been rejected in favor of EDARIC with selective deletion for all the performance measures, i.e. accuracy, kappa and g-mean with level of significance α= 0.05.

C. Comparison with other works

Table V summarizes the comparison of EDARIC and other state-of-the-art LCS algorithms. A brief description of other algorithms can be found in [14]. The average performance in the table are the mean of the individual performances over 20 data-sets which are in turn the average of 25 runs over

TABLE IV: Comparison of Results between EDARIC with selective deletion and random deletion

Dataset	Standard Datasets				Dataset	Imbalanced Datasets	
	Selective deletion		Random deletion			Selective deletion	Random deletion
	Tst acc	Tst kappa	Tst acc	Tst kappa		Tst g-mean	Tst g-mean
australian	87.82 ± 1.26	0.752 ± 0.0262	86.54 ± 1.77	0.725 ± 0.0369	abalone9-18	69.62 ± 8.87	70.52 ± 10.71
cleveland	60.13 ± 3.03	0.3324 ± 0.0674	58.57 ± 1.98	0.3174± 0.0528	glass0	85.49 ± 4.07	84.63 ± 4.75
crx	88.49 ± 3.0	0.767 ± 0.0613	86.73 ± 3.14	0.7313 ± 0.0646	glass2	67.24 ± 9.62	64.19 ± 8.44
heart	85.28 ± 5.3	0.6994 ± 0.1107	83.39 ± 6.03	0.662 ± 0.1241	glass5	94.86 ± 2.25	92.94 ± 4.32
hepatitis	93.86 ± 6.23	0.6969 ± 0.3779	92.26 ± 4.87	0.6229 ± 0.3434	haberman	62.99 ± 4.99	60.06 ± 3.52
lymphography	85.24 ± 5.64	0.7149 ± 0.1097	78.89 ± 6.91	0.5903 ± 0.1375	pima	77.99 ± 3.85	76.22 ± 3.92
pima	79.13 ± 2.57	0.5154 ± 0.0616	78.43 ± 2.39	0.5004 ± 0.0575	wisconsin	97.73 ± 0.38	97.28 ± 0.71
					yeast2	69.8 ± 2.78	67.0 ± 3.17
					yeast5	95.93 ± 2.3	95.83 ± 2.34

each individual data-set. The results of other algorithms were collected from [14].

Table V demonstrates that EDARIC is better compared to other algorithms in case of both standard and imbalanced problems. In terms of accuracy, kappa and g-mean, our algorithm outperforms all other algorithms by a significant margin, from which the superiority of EDARIC is clearly evident. As expected, results obtained by EDARIC in case of imbalanced problems are more attractive than in case of balanced ones.

It is evident from Table V that the testing performance of our algorithm does not vary too much with respect to the training performance. This is true for both balanced and imbalanced classification problems. For example, the average training accuracy and the testing accuracy achieved by our algorithm for the standard problems were 84.02% and 79.11% respectively. For the imbalanced problems, average training G-mean and testing G-mean achieved by our algorithm were respectively 85.55% and 83.99%. All these results indicate one point i.e., the techniques incorporated in EDARIC work nicely in handling two prominent difficulties, over-fitting and class-imbalance.

For a pair-wise comparison between EDARIC and other algorithms, Wilcoxon signed rank test [28] was used. Table VI shows the summary of pair-wise Wilcoxon signed rank test between EDARIC and other LCS approaches for both standard and imbalanced problems. Here, R^+ corresponds to the sum of ranks for EDARIC and R^- for the compared algorithm. These results show that null hypothesis has been rejected in favor of EDARIC against all the compared evolutionary algorithms with a significance level 0.05.

We also compare EDARIC with state-of-the-art non-evolutionary techniques, namely, CART [7], AQ [21], CN2 [10], C4.5 [24], C4.5 Rules [25] and Ripper [11] for both standard and imbalanced problems. These results are shown in Table VII. The corresponding Wilcoxon signed rank test summary is in shown Table VIII. Wilcoxon signed rank test rejects the null hypothesis in favor of EDARIC in all the possible cases.

For better visualization, we show the box-plot and star-plot of accuracy (in case of standard data-sets) for EDARIC and other evolutionary as well as non-evolutionary approaches in Figs. 3 and 4 respectively. Similarly, box-plot and star-plot of g-mean (in case of imbalanced data-sets) are shown in Figs. 5 and 6 respectively. The box-plots demonstrate the median, extreme values, quartiles etc. of the accuracy results. Star plots represent the performance as the distance from the center; hence a higher area determines better average performance. Both the box-plots and star-plots demonstrate the superiority of the proposed method.

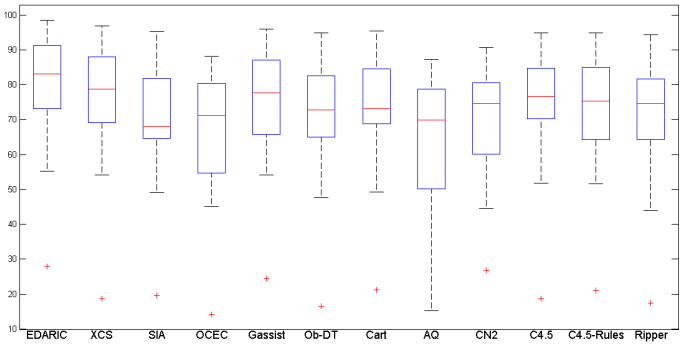


Fig. 3: Box plot for Accuracy to compare EDARIC and other evolutionary and non-evolutionary approaches

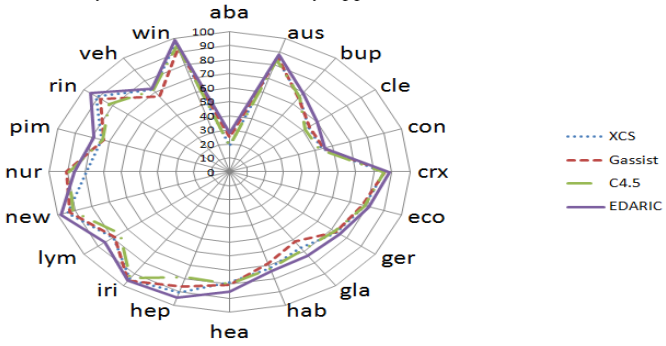


Fig. 4: Star plot of Accuracy to compare EDARIC and other evolutionary and non-evolutionary approaches

IV. Conclusion

In this paper, we propose a new evolutionary system, EDARIC, for rule extraction and classification. The idea behind EDARIC is to put more emphasis on generality of rules, rather than on specificity. The benefit of generality is avoidance (or minimization) of over-fitting suffered. A number of techniques have been adopted in our system to produce more general rules. The intelligent batch deletion mechanism is one such example which has ability to gradually convert more specific rules to more general ones. Furthermore, evolution of

TABLE V: Comparison of results between EDARIC and other state-of-the-art evolutionary approaches

		Standard Datasets						Imbalanced Datasets		
		Accuracy			Kappa			Gmean		
Family	**Algorithm**	**Training**	**Test**	**Rank**	**Training**	**Test**	**Rank**	GM_{trn}	GM_{tst}	**Rank**
Michigan	XCS [29]	88.40 ± 2.64	75.39 ± 4.12	4.15 (2)	0.80 ± 0.05	0.55 ± 0.08	4.55 (3)	60.62 ± 10.30	47.97 ± 9.72	8.15 (9)
	UCS [6]	92.10 ± 2.94	72.65 ± 5.58	7.35 (5)	0.84 ± 0.05	0.49 ± 0.11	7.25 (4)	72.36 ± 13.12	59.00 ± 16.55	6.55 (5)
IRL	SIA [27]	99.09 ± 0.23	69.79 ± 3.58	8.60 (7)	0.98 ± 0.00	0.42 ± 0.06	8.35 (7)	99.28 ± 0.38	63.85 ± 9.49	5.95 (4)
	HIDER [1]	85.43 ± 1.14	69.20 ± 3.64	8.60 (7)	0.70 ± 0.02	0.42 ± 0.07	9.05 (9)	55.12 ± 7.94	47.46 ± 11.05	9.20 (10)
GCCL	CORE [26]	69.64 ± 3.29	66.92 ± 4.64	11.50 (13)	0.37 ± 0.07	0.07 ± 0.09	13.05 (16)	52.81 ± 10.26	47.72 ± 14.74	9.60 (11)
	OCEC [20]	78.57 ± 3.28	67.08 ± 4.67	11.00 (12)	0.62 ± 0.05	0.43 ± 0.08	9.40 (10)	76.24 ± 3.74	67.10 ± 10.38	8.05 (8)
	COGIN [17]	85.71 ± 1.59	63.31 ± 4.35	11.80 (14)	0.71 ± 0.03	0.37 ± 0.06	11.90 (14)	53.49 ± 10.65	40.62 ± 16.13	11.85 (15)
Pittsburgh	EDARIC	84.02 ± 1.08	**79.11 ± 3.09**	**1.30 (1)**	0.72 ± 0.02	**0.62 ± 0.07**	**1.30 (1)**	85.55 ± 1.54	**83.99 ± 4.13**	**1.10 (1)**
	GIL [19]	77.37 ± 3.73	65.72 ± 5.19	12.45 (16)	0.62 ± 0.05	0.41 ± 0.08	10.45 (11)	75.92 ± 3.60	67.46 ± 9.87	7.45 (7)
	Pitts-GIRLA [12]	77.91 ± 2.81	63.71 ± 10.10	11.85 (15)	0.61 ± 0.04	0.31 ± 0.15	12.50 (15)	48.24 ± 10.61	30.32 ± 17.52	12.45 (16)
	DMEL [2]	44.74 ± 5.84	42.89 ± 6.06	15.60 (17)	0.20 ± 0.06	0.12 ± 0.06	15.55 (17)	20.48 ± 10.26	18.26 ± 12.07	13.90 (17)
	GASSIST [3]	84.06 ± 1.28	74.62 ± 3.71	4.60 (3)	0.71 ± 0.02	0.53 ± 0.07	4.35 (2)	68.33 ± 5.02	57.15 ± 9.69	6.65 (6)
	OIGA [30]	78.13 ± 2.41	71.51 ± 4.14	7.70 (6)	0.57 ± 0.05	0.47 ± 0.08	8.40 (8)	59.61 ± 5.65	49.48 ± 10.65	9.95 (12)
	ILGA [18]	75.50 ± 2.74	69.53 ± 4.28	9.35 (9)	0.53 ± 0.05	0.43 ± 0.08	10.50 (12)	53.87 ± 8.95	41.62 ± 11.12	11.60 (13)
HEDT	DT-GA [9]	80.12 ± 1.86	72.83 ± 3.39	7.30 (4)	0.64 ± 0.03	0.49 ± 0.06	7.35 (5)	75.44 ± 7.81	66.35 ± 11.60	5.25 (3)
	Oblique-DT [8]	99.73 ± 0.04	71.70 ± 3.34	9.60 (10)	0.99 ± 0.01	0.49 ± 0.06	8.15 (6)	99.94 ± 0.01	70.88 ± 8.55	4.85 (2)
	TARGET [16]	69.61 ± 2.42	67.95 ± 4.02	10.00 (11)	0.45 ± 0.05	0.39 ± 0.08	10.80 (13)	33.97 ± 14.14	33.09 ± 14.68	11.80 (14)

TABLE VI: Pair-wise wilcoxon signed rank test summary between EDARIC and other evolutionary approaches. R^+ corresponds to the sum of ranks for EDARIC and R^- for the compared algorithm. R in column Hypothesis(H) means null hypothesis is Rejected with $\alpha = 0.05$

		Standard Datasets								Imbalanced Datasets			
		Accuracy				Kappa				G-mean			
Family	**Algorithm**	R^+	R^-	**H**	**p-value**	R^+	R^-	**H**	**p-value**	R^+	R^-	**H**	**p-value**
Michigan	EDARIC Vs XCS [29]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
	EDARIC Vs UCS [6]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
IRL	EDARIC Vs SIA [27]	205	5	R	0.000189	204	6	R	0.000219	206	4	R	0.000163
	EDARIC Vs HIDER [1]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
GCCL	EDARIC Vs CORE [26]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
	EDARIC Vs OCEC [20]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
	EDARIC Vs COGIN [17]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
Pittsburgh	EDARIC Vs GIL [19]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
	EDARIC Vs Pitts-GIRLA [12]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
	EDARIC Vs DMEL [2]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
	EDARIC Vs GASSIST [3]	201	9	R	0.000338	202	8	R	0.000293	210	0	R	0.000089
	EDARIC Vs OIGA [30]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
	EDARIC Vs ILGA [18]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
HEDT	EDARIC Vs DT-GA [9]	209	1	R	0.000103	209	1	R	0.000103	209	1	R	0.000103
	EDARIC Vs Oblique-DT [8]	201	9	R	0.000338	202	8	R	0.000293	210	0	R	0.000089
	EDARIC Vs TARGET [16]	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089

TABLE VII: Comparison of EDARIC and state-of-the-art non-evolutionary approaches

	Standard Datasets						Imbalanced Datasets		
	Accuracy			Kappa			Gmean		
Algorithm	Training	Test	Avg.Rank	Training	Test	Avg.Rank	GM_{trn}	GM_{tst}	Avg.Rank
EDARIC	84.02 ± 1.08	**79.11 ± 3.09**	**1.15 (1)**	0.7260 ± 0.0251	**0.6200 ± 0.0758**	**1.15 (1)**	85.55 ± 1.54	**83.99 ± 4.13**	**1.15 (1)**
CART	85.17 ± 3.71	73.10 ± 3.91	3.85 (3)	0.7346 ± 0.0664	0.5070 ± 0.0744	3.80 (4)	79.69 ± 5.31	61.50 ± 11.51	4.35 (5)
AQ	79.27 ± 3.36	63.25 ± 5.18	6.25 (7)	0.6514 ± 0.0587	0.3554 ± 0.0971	6.25 (7)	60.32 ± 3.87	52.89 ± 8.72	5.90 (6)
CN2	79.95 ± 1.46	70.62 ± 3.51	4.85 (6)	0.6000 ± 0.0314	0.4143 ± 0.0746	5.80 (6)	49.36 ± 5.70	38.62 ± 12.09	6.30 (7)
C4.5	89.19 ± 1.47	73.98 ± 3.49	3.20 (2)	0.7911 ± 0.0362	0.5184 ± 0.0647	3.30 (2)	80.91 ± 6.01	69.62 ± 10.82	3.60 (4)
C4.5-Rules	81.43 ± 2.05	72.94 ± 4.10	3.90 (4)	0.6696 ± 0.0440	0.5079 ± 0.0779	3.75 (3)	81.13 ± 4.59	71.66 ± 10.23	3.15 (3)
Ripper	88.15 ± 2.42	70.81 ± 4.24	4.75 (5)	0.8023 ± 0.0387	0.5084 ± 0.0712	3.95 (5)	94.83 ± 0.86	74.14 ± 8.98	2.95 (2)

TABLE VIII: Wilcoxon signed rank test summary between EDARIC and state-of-the-art non-evolutionary approaches. R^+ corresponds to the sum of ranks for EDARIC and R^- for the compared algorithm. R in column Hypothesis(H) means null hypothesis is Rejected with $\alpha = 0.05$

	Standard Datasets								Imbalanced Datasets			
	Accuracy				Kappa				Gmean			
Algorithm	R^+	R^-	**H**	**p-value**	R^+	R^-	**H**	**p-value**	R^+	R^-	**H**	**p-value**
EDARIC Vs CART	209	1	R	0.000103	209	1	R	0.000103	210	0	R	0.000089
EDARIC Vs AQ	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
EDARIC Vs CN2	210	0	R	0.000089	210	0	R	0.000089	210	0	R	0.000089
EDARIC Vs C4.5	203	7	R	0.000254	204	6	R	0.000219	210	0	R	0.000089
EDARIC Vs C4.5-Rules	210	0	R	0.000089	210	0	R	0.000089	205	5	R	0.000189
EDARIC Vs Ripper	206	4	R	0.000163	205	5	R	0.000189	204	6	R	0.000219

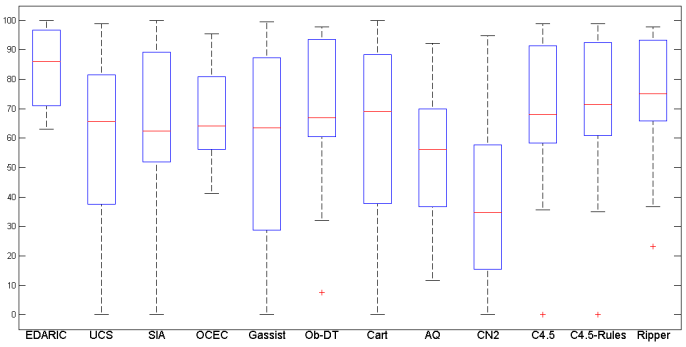


Fig. 5: Box plot for G-mean to compare EDARIC and other evolutionary and non-evolutionary approaches

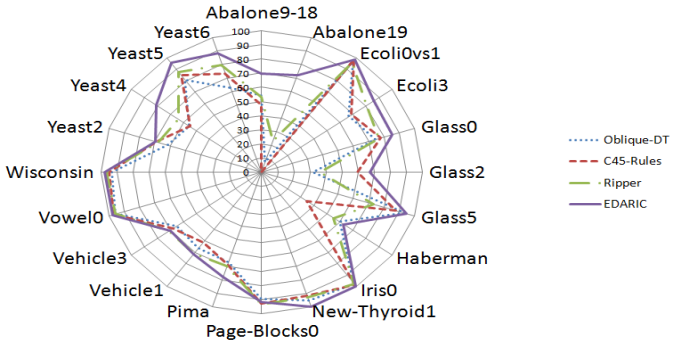


Fig. 6: Star plot of G-mean to compare EDARIC and other evolutionary and non-evolutionary approaches

multiple populations and ensemble with post-processing step in EDARIC helps in preserving valuable information about the minority classes for imbalanced problems.

EDARIC has been tested on 20 standard and 20 imbalanced benchmark classification data-sets obtained from UCI machine learning repository. Superior results have been produced by EDARIC in comparison with other algorithms as indicated in section III. In its current implementation, EDARIC has a few user-specified parameters although this is not unusual in the field. One of the future improvements to EDARIC would be to make these parameters adaptive. Another direction may be to design EDARIC in the light of multi-objective optimization concept.

Acknowledgments. The work has been done in the Computer Science & Engineering Department of Bangladesh University of Engineering and Technology (BUET). The authors would like to acknowledge BUET for its generous support.

REFERENCES

[1] J. S. Aguilar-Ruiz, R. Giraldez, and J. C. Riquelme. Natural encoding for evolutionary supervised learning. *Trans. Evol. Comp*, 11(4):466–479, Aug. 2007.

[2] W.-H. Au, K. C. Chan, and X. Yao. A novel evolutionary data mining algorithm with applications to churn prediction. *Trans. Evol. Comp*, 7(6):532–545, Dec. 2003.

[3] J. Bacardit and J. M. Garrell. Evolving multiple discretizations with adaptive intervals for a pittsburgh rule-based learning classifier system. In *Proceedings of the 2003 international conference on Genetic and evolutionary computation: PartII*, GECCO'03, pages 1818–1831, Berlin, Heidelberg, 2003. Springer-Verlag.

[4] J. Bacardit and J. M. Garrell. Bloat control and generalization pressure using the minimum description length principle for a pittsburgh approach learning classifier system. In *Proceedings of the 2003-2005 international conference on Learning classifier systems*, IWLCS'03-05, pages 59–79, Berlin, Heidelberg, 2007. Springer-Verlag.

[5] R. Bajcsy, editor. *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*. Morgan Kaufmann, 1993.

[6] E. Bernadó-Mansilla and J. M. Garrell-Guiu. Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evol. Comput.*, 11(3):209–238, Sept. 2003.

[7] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[8] E. Cantu-Paz and C. Kamath. Inducing oblique decision trees with evolutionary algorithms. *Trans. Evol. Comp*, 7(1):54–68, Feb. 2003.

[9] D. R. Carvalho and A. A. Freitas. A hybrid decision tree/genetic algorithm method for data mining. *Inf. Sci.*, 163(1-3):13–35, June 2004.

[10] P. Clark and T. Niblett. The cn2 induction algorithm. *Mach. Learn.*, 3(4):261–283, Mar. 1989.

[11] W. W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[12] A. L. Corcoran and S. Sen. Using real-valued genetic algorithms to evolve rule sets for classification. In *In IEEE-CEC*, pages 120–124, 1994.

[13] J. Demar, B. Zupan, G. Leban, and T. Curk. Orange: From experimental machine learning to interactive data mining. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Knowledge Discovery in Databases: PKDD 2004*, volume 3202 of *Lecture Notes in Computer Science*, pages 537–539. Springer Berlin Heidelberg, 2004.

[14] A. Fernández, S. García, J. Luengo, E. Bernadó-Mansilla, and F. Herrera. Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study. *Trans. Evol. Comp*, 14(6):913–941, Dec. 2010.

[15] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

[16] J. B. Gray and G. Fan. Classification tree analysis using target. *Comput. Stat. Data Anal.*, 52(3):1362–1372, Jan. 2008.

[17] D. P. Greene and S. F. Smith. Competition-based induction of decision models from examples. *Mach. Learn.*, 13(2-3):229–257, Nov. 1993.

[18] S.-U. Guan and F. Zhu. An incremental approach to genetic-algorithms-based classification. *Trans. Sys. Man Cyber. Part B*, 35(2):227–239, Apr. 2005.

[19] C. Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Mach. Learn.*, 13(2-3):189–228, Nov. 1993.

[20] L. Jiao, J. Liu, and W. Zhong. An organizational coevolutionary algorithm for classification. *Trans. Evol. Comp*, 10(1):67–80, Sept. 2006.

[21] R. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The Multi-Purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, 1986.

[22] A. Orriols-Puig and E. Bernad-Mansilla. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing*, 13(3):213–225, 2009.

[23] A. Orriols-Puig, E. Bernadó-Mansilla, D. E. Goldberg, K. Sastry, and P. L. Lanzi. Facetwise analysis of xcs for problems with class imbalances. *Trans. Evol. Comp*, 13(5):1093–1119, Oct. 2009.

[24] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[25] J. R. Quinlan. Mdl and categorical theories (continued). In *In Machine Learning: Proceedings of the Twelfth International Conference, Lake Taho*, pages 464–470. Morgan Kaufmann, 1995.

[26] K. C. Tan, Q. Yu, and J. H. Ang. A coevolutionary algorithm for rules discovery in data mining. *International Journal of Systems Science*, 37(12):835–864, 2006.

[27] G. Venturini. Sia: A supervised inductive algorithm with genetic search for learning attributes based concepts. In *Proceedings of the European Conference on Machine Learning*, ECML '93, pages 280–296, London, UK, UK, 1993. Springer-Verlag.

[28] F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[29] S. W. Wilson. Classifier fitness based on accuracy. *Evol. Comput.*, 3(2):149–175, June 1995.

[30] F. Zhu and S.-U. Guan. Ordered incremental training with genetic algorithms. *Int. J. Intell. Syst.*, 19(12):1239–1256, Dec. 2004.