Genetic learning of fuzzy rule bases for multi-label classification using an iterative approach

Edward Hinojosa Cárdenas
Department of Computer Science
Universidad Nacional de San Agustín de Arequipa
Arequipa, Peru
ehinojosa@unsa.edu.pe

Edgar Sarmiento Calisaya
Department of Computer Science
Universidad Nacional de San Agustín de Arequipa
Arequipa, Peru
esarmientoca@unsa.edu.pe

Heloisa de Arruda Camargo
Department of Computer Science
Federal University of São Carlos
São Paulo, Brasil
heloisa@dc.ufscar.br

***Abstract*—Multi-label classification problems exist in many real world applications where to each example in the dataset can be assigned a set of target labels. This paper presents a new two-step method for genetic learning of a fuzzy rule base for multi-label classification, called IRL-MLC. The first step uses a genetic algorithm based on an iterative approach to learn a preliminary rule base where the fitness of each rule depends on the degree of firing calculated for the set of labels of each example (positive or negatively) in the dataset. The second step uses a genetic algorithm to tune weights of each fuzzy rule in the preliminary rule base where the fitness of each set of weights is the precision of the multi-label classification. Experiments are conducted on five multi-label datasets, in biology, multimedia and text domains, and the proposed method has been compared with one state-of-art method. Results provide interesting insights into the quality of the discussed novel method.**

***Index Terms*—Multi-label classification, genetic algorithm, iterative approach**

I. Introduction

Multi-label classification (MLC) is a predictive data mining task which consists of assigning, to each example, a set of target labels. MLC exists in many real world applications, for example: images [1], texts [2], video [3], audio [4], among others.

MLC methods are divided into two groups, problem transformation methods and algorithm adaptation methods. In problem transformation methods, the MLC problem is transformed into single-label classification problem and the classification process follow the single-label classification; in algorithm adaption methods, a single-label algorithm is modified to applied on multi-label data. In [5] the authors show a comparative study of most common multi-label classification methods: copy, copy-weight select-max, select-min, select-random, ignore, binary relevance, label power set and classifier chain, belong to the group of problem transformation methods; and, multi-label decision-tree [6], multi-Label k nearest neighbors [7], support vector machine with Heterogeneous feature kernel [8], ranking support vector machine [9], multi-label tree based boosting methods [10], multi-label neural networks [11] and multi-label naive bayesian [12], belong to the group of algorithm adaptation methods.

Fuzzy Systems (FS) and the most common fuzzy model called Fuzzy Rule-Base Systems (FRBS) have been successfully applied to solve different problems in a variety of domains; when sufficient amounts of data are available, it is advantageous to build such systems automatically employing methodologies from different machine learning paradigms such as Evolutionary Algorithms (EA). Evolutionary Fuzzy Systems (EFS) are a successful hybridization between FS and EA, they integrate both the management of imprecision/uncertainty and inherent interpretability of FRBS, with the learning and adaptation capabilities of evolutionary optimization [13].

The nature of the multi-label problem is well suited for the use of FRBS and EFS based algorithms and MLC has been cited as one of the promissing trends for EFS in [13]. However, when searching the literature for articles that address the problem of learning fuzzy systems for MLC problems, the authors of this article identified that there are not enough efforts being devoted to the topic. To the best of our knowledge, only two research works to learning fuzzy rules for MLC were found and both belong to the problem transformation method group; the first one was proposed in [14], where the author evaluates Fuzzy Unordered Rule Induction Algorithm (FURIA) [15] as the base classifier after using different strategies to transform a multi-label problem into a different single-label classification problems; the second one was proposed in [16], where the strategy Generalized k-Labelsets Ensemble (GLE) [17], which performs the basis expansion method to train Label powerset classifier on random k label sets, is used to transform a multi-label problem into a different single-label classification problems before to apply FURIA for learning a fuzzy rule base. No articles could be found in the algorithms adaptation methods group.

In the last years, our research has focussed on algorithms

978-1-7281-6932-3/20/$31.00 ©2020 IEEE

to learn or tune components of FRBS using GA either in the traditional or multiobjective versions for single-label classification [18], [19], [20], [21]. This paper proposes a new method to learn a Rule Base (RB) for a fuzzy multi-label classifier using two genetic algorithms (GA). The first GA is based on the iterative approach where the GA is run several times, each time generating a single rule for learning a preliminary RB. The iterative approach has been explored before in our research [20], [21] and demonstrated to be efficent for the generation of rule bases. The second GA tunes the preliminary RB for improving the hamming loss [22] in a single run.

The evolution process of the first GA starts after three previous steps: 1) to define uniformly distributed fuzzy sets for all attributes that describe the examples in the dataset; 2) to generate an initial population or parent population in the first iteration where each chromosome represents a fuzzy if-then rule such that each gene in the chromosome represents a fuzzy antecedent and the last gen represents a fuzzy rule class; and 3) to calculate the fitness of each chromosome by means of the subtraction between the sum of positive degrees of firing and sum of negatives degree of firing of the rule. The evolution process (selection, crossover and mutation operators) generates an offspring population. Next, the current parent population and the offspring population are merged and the next parent population is generated by adding the fuzzy rules with maximum fitness selected from the merged population. At the end of the evolution process, the best fuzzy rule in the last parent population is added to the preliminary RB. The first GA is run a defined number of times which is the number of rules in the preliminary RB.

The second GA tunes the fuzzy rule weights for improving the preliminary RB hamming loss. The fitness calculation, which is the main characteristic of this GA, considers a threshold for assigning one or more classes to an example. The threshold is obtained from the average of the sum of firing degrees of all fuzzy rules for each class. The evolution process is similar to the first GA with relation to the genetic operators application and the generation of the next population except that the chromosomes selected for the next population are the ones with the lowest fitness values. At the end of the evolution process, the best chromosome in the last parent population, which is the best combination of rules weights, is selected and, these weights, with their corresponding rules in the preliminary RB, define the final RB.

The next section explains briefly the multi-label classification problem. In Section III the proposal of the method to genetic learning of fuzzy rule bases for multi-label classification based on an iterative approach is detailed. Section IV describes the experiments and discusses the results of the proposal. Finally, Section V contains conclusions and future works.

II. Multi-label Classification

Multi-label learning is a supervised learning paradigm that recently has been investigated in an increasing number of fields where it can be applied in domains as mentioned before. All these applications have in common that, for each example in the dataset, one or more labels can be assigned, as shows the example in Table I.

TABLE I
DATASET WITH MULTI-LABEL EXAMPLES

Ex.	**Atributes** (A)			**Labels** (L)				**Example labels**
(E)	A_1	A_2	A_3	l_1	l_2	l_3	l_4	(Y)
e_1	a_{1_1}	a_{1_2}	a_{1_3}	0	0	1	1	$Y_1 = \{L_3, L_4\}$
e_2	a_{2_1}	a_{2_2}	a_{2_3}	0	0	1	0	$Y_2 = \{L_3\}$
e_3	a_{3_1}	a_{3_2}	a_{3_3}	1	1	0	0	$Y_3 = \{L_1, L_2\}$
e_4	a_{4_1}	a_{4_2}	a_{4_3}	1	0	1	1	$Y_4 = \{L_1, L_3, L_4\}$

An important task in multi-label learning is the Multi-Label Classification (MLC). The problem of multi-label classification consists in predicting labels for a multi-label dataset D. Each element of D is a pair $D_i = (e_i, Y_i)$, where e_i is an example of a set of examples $E = \{e_i : i = 1 \dots v\}$ with v being the number of examples and Y_i is a subset of the set of labels $L = \{l_k : k = 1 \dots q\}$ with q being the maximum number of labels. Each example e_i is described by n attributes whose values are defined as $A_j = \{a_{i_j} : j = 1 \dots g\}$). For a multi-label classification problem $|Y_i| \geq 2$ in, at least, one element of D.

Methods for MLC can be divided into two groups: problems transformation methods and algorithms adaptation methods, both are detailed in subsection II-A and II-B, respectively [23].

A. Problem transformation methods

Problem transformation methods preprocess the dataset to convert a multi-label data set to a single-label dataset with the same set of labels. A single-label classifier that outputs probability distributions over all classes can then be used to learn a ranking. The class with the highest probability will be ranked first, the class with the second best probability will be ranked second, and so on.

The most common transformations methods are simple transformations (copy, copy-weight, select-max, select-min, select-random (one of the possible) and ignore), binary relevance, label power set, classifier chain, include labels classifier, among others [24], [25].

B. Algorithm adaptation methods

Algorithm adaptation methods handle directly the multi-label classification problem using adaptations of several algorithms, for example, multi-label decision-tree [6], multi-Label k nearest neighbors [7], support vector machine with Heterogeneous feature kernel [8], ranking support vector machine [9], multi-label tree based boosting methods [10], multi-label neural networks [11] and multi-label naive bayesian [12], among others.

The method proposed in this paper is an algorithm adaption method for the genetic learning of a FRBS to address directly the multi-label classification problem, but it preprocesses a dataset in in a way similar to the simple transformation copy-weight, where the weight of each label of example e_i is

calculated by $\frac{1}{|Y_i|}$. An example of that transformation (based on Table I) is shown in Table II.

TABLE II
SIMPLE TRANSFORMATION: COPY-WEIGHT

Example	Labels	Weight
(E)	(L)	W
$e_1(a)$	L_3	0.5
$e_1(b)$	L_4	0.5
e_2	L_3	1.0
$e_3(a)$	L_1	0.5
$e_3(b)$	L_2	0.5
$e_4(a)$	L_1	0.33
$e_4(b)$	L_3	0.33
$e_4(c)$	L_4	0.33

Both problem transformation and algorithm adaptation methods generate a multi-label classifier which should be evaluated. There are some suitable metrics to evaluate multi-label classifiers, the subsection II-C detailed the metrics used in this paper [23].

C. Performance Metrics

1) Hamming Loss: defines as the proportion of labels whose relevance is incorrectly predicted:

$$Hamming(Y,P) = \frac{1}{v * q}\sum_{i=1}^{v}\sum_{k=1}^{q}[\![Y_{ik} \neq P_{ik}]\!] \quad (1)$$

2) Accuracy: computes the percentage of relevant labels predicted in the subset formed by the union the returned and relevant labels:

$$Accuracy(Y,P) = \frac{\sum_{i=1}^{v}\sum_{k=1}^{q}[\![Y_{ik} = 1 \ and \ P_{ik} = 1]\!]}{\sum_{i=1}^{v}\sum_{k=1}^{q}[\![Y_{ik} = 1 \ or \ P_{ik} = 1]\!]} \quad (2)$$

3) Precision: , determines the fraction of relevant labels in the predicted labels:

$$Precision(Y,P) = \frac{\sum_{i=1}^{v}\sum_{k=1}^{q}[\![Y_{ik} = 1 \ and \ P_{ik} = 1]\!]}{\sum_{i=1}^{v}\sum_{k=1}^{q}[\![P_{ik} = 1]\!]} \quad (3)$$

4) Recall: is the proportion of relevant labels correctly predicted:

$$Recall(Y,P) = \frac{\sum_{i=1}^{v}\sum_{k=1}^{q}[\![Y_{ik} = 1 \ and \ P_{ik} = 1]\!]}{\sum_{i=1}^{v}\sum_{k=1}^{q}[\![Y_{ik} = 1]\!]} \quad (4)$$

5) F_1: is the evenly weighted harmonic mean of Precision and Recall:

$$F_1(Y,P) = \frac{2 \times \sum_{i=1}^{v}\sum_{k=1}^{q}[\![Y_{ik} = 1 \ and \ P_{ik} = 1]\!]}{\sum_{i=1}^{v}\sum_{k=1}^{q}([\![Y_{ik} = 1]\!] + [\![P_{ik} = 1]\!])} \quad (5)$$

In all equations above Y is the set of real labels for each example; Y_{ik} is the label value l_k of example e_i, 1 if the label l_k is assigned to e_i, 0 otherwise; P is the set of predicted labels for the multi-label classifier for each example; P_{ik} is the label value l_k of example e_i, 1 if the label l_k is predicted to e_i, 0 otherwise; and $[\![pred]\!]$ evaluates to 1 if the predicate $pred$ is true and 0 otherwise [26].

III. PROPOSED METHOD

This section details the proposed two step method for learning fuzzy rule bases for multi-label classification, called Iterative Rule Learning to Multi-label Classification (IRL-MLC). As mentioned above, the first step uses a GA based on an iterative approach for learning a preliminary RB; the second step uses a GA for tuning the weights of fuzzy rules to improve the precision of the preliminary RB to obtain the final RB. Each step of the proposed method is detailed below.

A. Learning the Preliminary RB

This step is based on a genetic iterative approach where each chromosome represents only one fuzzy rule and the GA is executed several times. At the end of each execution of the GA, the chromosome with the highest fitness value, which is the best fuzzy rule, is added to the RB. This process is repeated until the RB is full. Before each execution of the GA, the dataset is updated by changing the weights of the data. The required parameters are the population size ps and the RB size rbs. Figure 1 shows the flowchart of the genetic iterative approach.

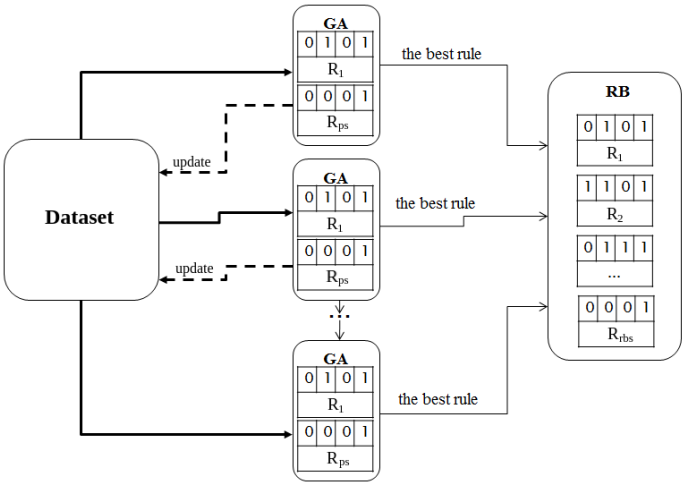

Fig. 1. Flowchart of Genetic Iterative Approach

Before the genetic learning of preliminary RB, the dataset is preprocessed by means of calculating label weights of each example assigning equal weights to each label, such that they sum up to 1. An example of the preprocessing step is shown in Table III (based on Table I). Those weights are important for calculating the fitness of each fuzzy rule for the GA.

TABLE III
PREPROCESS OF DATASET

Ex.	Atributes (A)			Labels (L)			
(E)	A_1	A_2	A_3	lw_1	lw_2	lw_3	lw_4
e_1	a_{1_1}	a_{1_2}	a_{1_3}	0.00	0.00	0.50	0.50
e_2	a_{2_1}	a_{2_2}	a_{2_3}	0.00	0.00	1.00	0.00
e_3	a_{3_1}	a_{3_2}	a_{3_3}	0.5	0.5	0.00	0.00
e_4	a_{4_1}	a_{4_2}	a_{4_3}	0.33	0.00	0.33	0.33

The genetic iterative learning of preliminary RB is based on the standard GA [27]. Figure 2 shows the flowchart of this step. The details of each sub-step are described below.

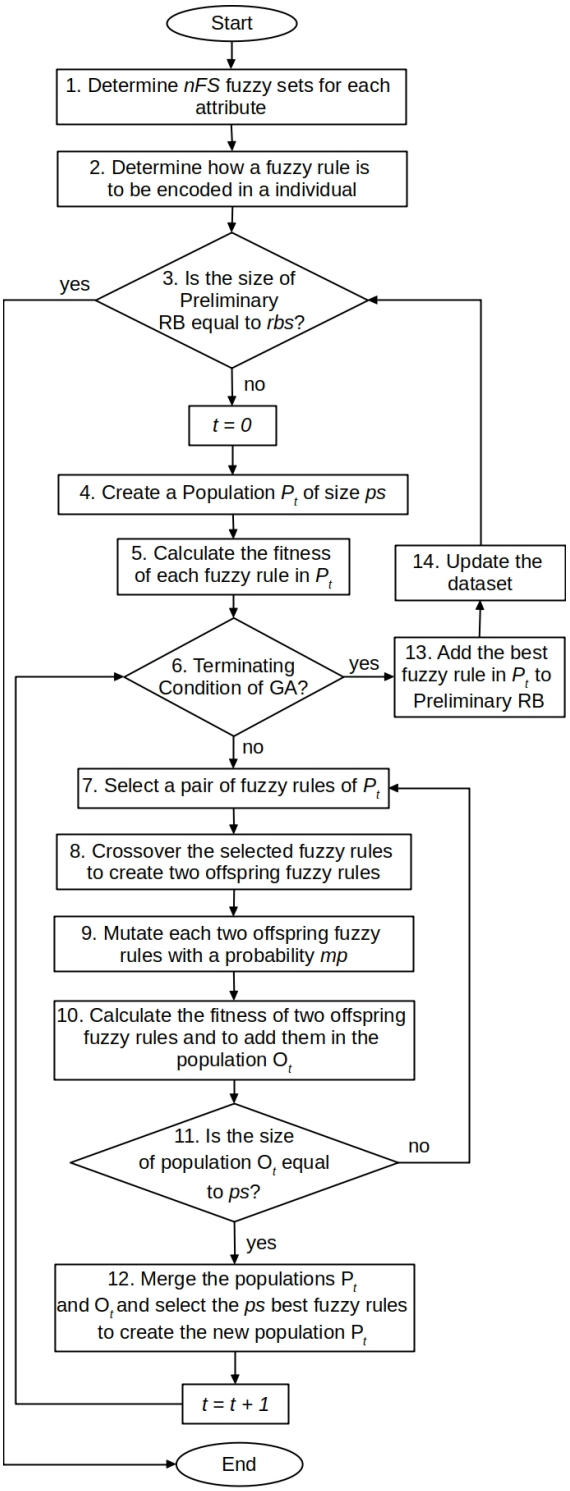

Fig. 2. Flowchart of Genetic Iterative Learning of Preliminary RB

1) Determining fuzzy sets for each attribute: For each attribute the minimum (min) and maximum value (max) are obtained. After, nFS triangular fuzzy sets are defined uniformly distributed on the attribute domain, i.e., each fuzzy set has the same support and they cover all range between values maximum and minimum.

Figure 3 shows an example with $nFS = 5$.

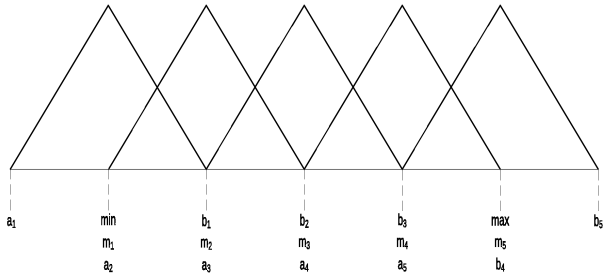

Fig. 3. Triangular Fuzzy Sets Uniformly Distributed for an Attribute

In case the attributes are nominal (1 or 0) just a triangular fuzzy set is defined where the membership function is 1.0 for value 1 and 0.0 for value 0.

2) Encoding a fuzzy rule: This step uses a decimal encoding with three parts. The first part represents the antecedent part of the fuzzy rule where each gene represents an index of a fuzzy set (or linguistic term) for each attribute (value zero represents a *don't care* condition, what means that the respective attribute does not appear in the rule). The second part (only one gene) represents the class or consequent of the fuzzy rule. Finally, the third part (only one gene) represents the fitness of the fuzzy rule. Figure 4 illustrates the chromosome representation used in this step.

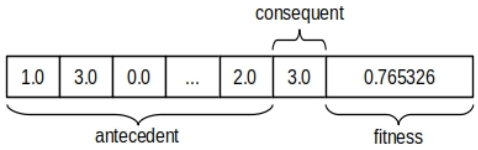

Fig. 4. Encoding a Fuzzy Rule

3) Size of Preliminary RB: The preliminary rule base size (rbs) is a parameter of the proposed method.

4) Creating the Initial Population: Before creating the initial population two parameters must be defined: the population size (ps) or the number of fuzzy rules in each iteration of the GA; and, the probability of *don't care* condition for each fuzzy rule (pdc).

The initial population is created by generating the fuzzy rules one by one. For each fuzzy rule generation, an example of the training dataset is selected randomly. After that, a random value is generated and, if it is greater than pdc, the gene value is defined as the index of the fuzzy set with the maximum membership degree of the value of the corresponding attribute of the selected example. If the random value is less than pdc, value zero is assigned to the antecedent gene.

Finally, a random index label with weight greater than zero of the example selected is assigned to consequent of the fuzzy rule and the fuzzy rule is added to the initial population. If the selected example doesn't have weight labels greater than zero a new random example is selected and all the process is repeated to generate a new fuzzy rule.

5) Calculating the fitness: The fitness of a fuzzy rule n ($fitness(fr_n)$) is calculated by means of the subtraction between the positive degree of firing ($df(fr_n)^+$) and the negative degree of firing ($df(fr_n)^-$). A fuzzy rule n is better that another fuzzy rule m if $fitness(fr_n) > fitness(fr_m)$)

$$fitness(fr_n) = df(fr_n)^+ - df(fr_n)^- \tag{6}$$

The $df(fr_n)^+$ is calculated by summing the degrees of firing of fuzzy rule n for each example i in the dataset (df_{ni}) times the label weight of label k of the example i (lw_{ik}), if lw_{ik} is greater than zero.

$$\begin{array}{l} df(fr_n)^+ = \sum_{i=1}^{v} df(fr_{ni}) \times lw_{ik} \\ (if\ k = cfr_n\ and\ lw_{ik} > 0.0) \end{array} \tag{7}$$

The $df(fr_n)^-$ is calculated by summing the degrees of firing of fuzzy rule n for each example i in the dataset (df_{ni}), if lw_{ik} is less than or equal to zero.

$$\begin{array}{l} df(fr_n)^- = \sum_{i=1}^{v} df(fr_{ni}) \\ (if\ k = cfr_n\ and\ lw_{ik} \leq 0.0) \end{array} \tag{8}$$

The degree of firing of a rule n for example i ((df_{ni})) is the minimum membership degree of the attribute values of example i to the corresponding antecedents of fuzzy rule n. antecedent *don't care* have value 1.0 as membership degree to the corresponding antecedent of the fuzzy rule n. A low negative value is assigned to the fitness of a fuzzy rule with all antecedents *don't* care.

6) Terminating condition of GA: The terminating condition of GA is the maximum number of iterations ni.

7) Selection operator: The tournament selection [28] is executed twice to selected two fuzzy rule parents. Each time ts fuzzy rules are selected randomly and the best fuzzy rule (depending the fitness) is chosen.

8) Crossover operator: The crossover has 100% of probability of being applied and is based on the uniform crossover to create two offspring fuzzy rules. Antecedents and consequent are copied randomly from the pair of parents fuzzy rules to two offspring fuzzy rules.

9) Mutation operator: The mutation operator is performed on each offspring fuzzy rules with a mutation probability (mp). In the mutation process a randomly generated fuzzy set index is set to each antecedent of the offspring fuzzy rule with a probability mp_a. The consequent of offspring fuzzy rule doesn't suffer a mutation process.

10) Calculating the fitness of offspring population: The fitness of the offspring fuzzy rules, mutated or not, is calculated based on equation 6 before it is added to the population O_t.

11) Reaching the size of offspring population: The evolution process is repeated until the number of fuzzy rules in the offspring population O_t is equal to ps.

12) Generating the new population: After the size of offspring population is equal to ps, the population O_t is merged to current population P_t and the ps best fuzzy rules, of the merged population, are selected to generate the next population P_t.

13) Selecting the best fuzzy rule: When the maximum number of iterations ni is reached the best fuzzy rule fr_b of the last population P_t is selected and added to the preliminary RB.

14) Updating the dataset: When the best fuzzy rule fr_b is added to the preliminary RB the dataset is updated before the next execution of the GA begins. The update is performed on lw_{ik} where index k is equal to the consequent or class of added fuzzy rule (cfr_b). The new lw_{ik} is equal to previous lw_{ik} minus the degree firing of the added fuzzy rule with the example i. The dataset update is a kind of punishment on the examples that are covered by the last rule generated, so that a different fuzzy rule, covering a different subset of the examples is able to be generated in the next GA execution.

$$\begin{array}{l} lw_{ik} = lw_{ik} - df(fr_{bi}) \\ (if\ k = cfr_b) \end{array} \tag{9}$$

For example, based on Table III, Table IV shows how some of the label weights with index $k = 3$ are decreased after a fuzzy rule with $cfr_b = 3$ was added to the preliminary RB.

TABLE IV
UPDATED DATASET WITH $k = cfr_b = 3$

Ex.	**Atributes** (A)			**Labels** (L)			
(E)	A_1	A_2	A_3	lw_1	lw_2	lw_3	lw_4
e_1	a_{1_1}	a_{1_2}	a_{1_3}	0.00	0.00	0.50	0.50
e_2	a_{2_1}	a_{2_2}	a_{2_3}	0.00	0.00	0.94	0.00
e_3	a_{3_1}	a_{3_2}	a_{3_3}	0.5	0.5	0.00	0.00
e_4	a_{4_1}	a_{4_2}	a_{4_3}	0.33	0.00	0.28	0.33

All the sub-steps detailed above are repeated until the population size of preliminary RB srb has been reached.

B. Tuning the Preliminary RB

This step is based on a standard GA where each chromosome represents a set of fuzzy rule weights, a weight for each fuzzy rule in the preliminary RB, and a threshold used to define the predicted labels for each example. Figure 5 shows the flowchart of this step. The details of each sub-step are described below.

1) Encoding a set of fuzzy rule weights: This step uses a decimal encoding for a chromosome with three parts. The first part represents the fuzzy rule weights part where each gene represents a weight for each fuzzy rule in the preliminary RB (rbs). The second part (only one gene) represents the threshold used to define the predicted labels for each example. Finally, the third part (only one gene) represents the fitness of the set of fuzzy rule weights represented in the chromosome. Figure 6 illustrates the chromosome representation used in this step.

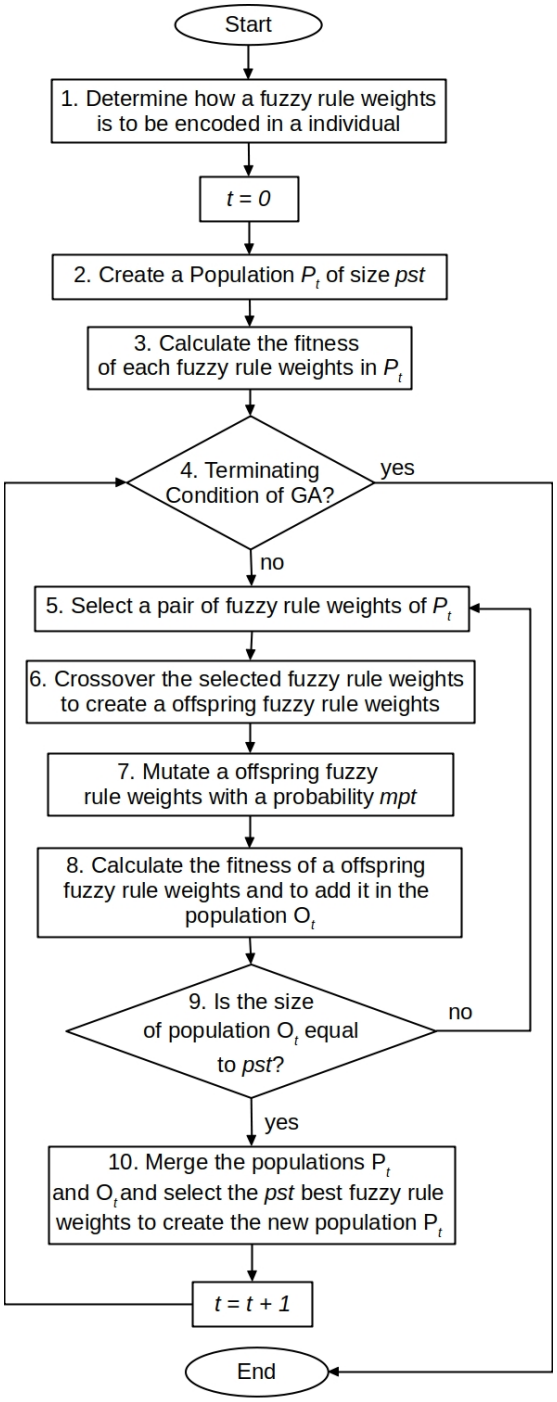

Fig. 5. Flowchart of Genetic Tuning of Preliminary RB

2) Creating the initial population: Before creating the initial population, the population size (pst) or number of set of fuzzy rule weights in each iteration of GA must be defined. The initial population is created by randomly generating the sets of fuzzy rule weights one by one (minimum and maximum value are 0.0 and 1.0 respectively). For each chromosome, a random threshold value is assigned to threshold part. The first fuzzy rule weights in the initial population has the value 1.0 for each weight and the value 0.0 for the threshold.

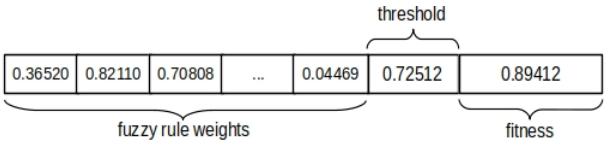

Fig. 6. Encoding a Fuzzy Rule Weights

3) Calculating the fitness: The fitness of a set of fuzzy rule weights z is calculated by means of the hamming loss between the real labels and predicted labels using the RB obtained in the previous step and the respective rules weights in the chromosome for each example. The calculus of Hamming loss is defined in Equation 1 ($hl(frw_z)$).

An important calculation in this step is how to define the predicted labels for each example. First, the degree of each label k for the example i (d_{ik}) is calculated by summing the firing degrees of each fuzzy rule n for example i ($df(fr_{ni})$) times the fuzzy rule weight z (frw_z) where k is equal to the consequent of rule n (cfr_n).

$$\begin{array}{l} d_{ik} = \sum_{n=1}^{rbs} df(fr_{ni}) \times frw_z \\ (if\ k = cfr_n) \end{array} \tag{10}$$

Second, the percentage degree of each label k for example i (pd_{ik}) is calculated by d_{ik} divided by the sum of each d_{ik} for all k labels.

$$pd_{ik} = \frac{d_{ik}}{\sum_{k=1}^{q} d_{ik}} \tag{11}$$

Finally, a label k is assigned to an example i if the pd_{ik} is greater than the threshold value.

4) Terminating condition of GA: The terminating condition of GA is the maximum number of iterations nit.

5) Selection operator: The tournament selection is executed twice. Each time, tst sets of fuzzy rule weights are selected randomly and the best one (the one with the lowest Hamming loss) is chosen.

6) Crossover operator: The crossover has 100% of probability of been applied and is based on the simulated binary crossover (SBX) to create an offspring fuzzy rule weights (considering the minimum and maximum values). The threshold value suffer the SBX too.

7) Mutation operator: The mutation operator is performed on an offspring fuzzy rule weights with a mutation probability (mpt). In the mutation process a random weight of the offspring fuzzy rule weights is changed to a random value between 0.0 and 1.0 with a probability mpt_wth. The threshold value doesn't suffer a mutation process.

8) Calculating the fitness of offspring population: The fitness of a chromosome, mutated or not, is calculated based on equation 1 and is detailed in sub-step III-B3.

9) Reaching the size of offspring population: The evolution process is repeated until the number of fuzzy rule weights in the offspring population O_t is equal to pst.

10) Generate the new population: After the size of offspring population is equal to pst, the population O_t is merged to current population P_t and the pst best fuzzy rule weights, of merged population, are selected to generate the next population P_t.

As mentioned above, the evolution process is repeated until the number of iterations nit is reached. The final set of fuzzy rule weights and the threshold value are the best set of fuzzy rule weights and threshold value in the last population P_t.

Finally, the final RB consists of three elements: all fuzzy rules obtained in the learned preliminary RB, the fuzzy rule weights and the threshold value obtained from the tuning process of the preliminary RB. The source code of the proposed method IRL-MLC is available on github (username: Edward-Hinojosa-Cardenas, project: IRL-MLC).

The next section presents the results obtained by the proposed method on two benchmark databases.

IV. Experimental analysis

The proposed method IRL-MLC has been tested on five benchmark datasets for evaluation: yeast, scene, corel5k, enron and medical. The datasets came from biology, multimedia and text domains and were obtained from KEEL multi-label dataset repository [29]. The specifications of the datasets are given in Table V: number of attributes (#Attributes - R=Real - N=Nominal), number of examples (#Examples), number of labels (#Labels).

TABLE V
Multi-label Dataset Specifications

Name	Domain	#Attributes	#Examples	#Labels
yeast	biology	103 (R)	2417	14
scene	multimedia	294 (R)	2407	6
corel5k	multimedia	499 (N)	5000	374
enron	text	1001 (N)	1702	53
medical	text	1449 (N)	978	45

The parameters for each step of the proposed method IRL-MLC, detailed in section III, are shown in Table VI.

TABLE VI
Parameters of Proposed Method IRL-MLC

Parameters of the genetic learning of preliminary RB (step 1)	
Number of fuzzy sets with real attributes (nFS)	5
Number of fuzzy sets with nominal attributes (nFS)	1
Preliminar rule base size (rbs)	400
Population size (ps)	200
Probability of *don't care* conditions (pdc)	0.9
Number of iterations of genetic algorithm (ni)	500
Number of fuzzy rules in tournament selection	2
Mutation probability (mp)	0.2
Parameters of the genetic tuning of preliminary RB (step 2)	
Population size (pst)	100
Number of iterations of genetic algorithm (nit)	1000
Number of fuzzy rule weights in tournament selection (tst)	2
SBX crossover (α)	0.5
Mutation probability (mpt)	0.1
Mutarion probability for each weight and threrhold (mpt_wth)	0.1

The results obtained by the proposed method IRL-MLC on the five datasets mentioned above for the five metrics defined in section II-C are given in Table VII. For the first metric (Hamming Loss), lower values indicate better results and for the other metrics (Accuracy, Precision, Recall and F_1) higher values indicate better results. The experiments were run using 10-fold cross validation and standard deviation is shown in parenthesis.

TABLE VII
Results of the proposed method IRL-MLC

Dataset	Hamming Loss	Accuracy	Precision	Recall	F_1
yeast	0.215 (0.004)	0.400 (0.008)	0.724 (0.019)	0.471 (0.010)	0.570 (0.009)
scene	0.113 (0.010)	0.502 (0.035)	0.707 (0.032)	0.634 (0.030)	0.668 (0.031)
corel5k	0.010 (0.000)	0.012 (0.003)	0.320 (0.069)	0.012 (0.003)	0.023 (0.006)
enron	0.058 (0.002)	0.146 (0.015)	0.688 (0.040)	0.157 (0.017)	0.255 (0.023)
medical	0.012 (0.001)	0.631 (0.036)	0.780 (0.024)	0.767 (0.039)	0.774 (0.028)

The performance of the proposed method IRL-MLC is compared with a state-of-art method called Online Sequential Multi-label Extreme Learning Machine (OSML-ELM) introduced in [30], based on the performance metrics mentioned above. The comparison results are given in Table VIII. For each metric, the best result is highlighted in bold and the difference between the results of the two mentioned methods is in the last column.

TABLE VIII
Comparison of the proposed method IRL-MLC and OSML-ELM method

Dataset	Metric	IRL_MLC	OSML-ELM	Difference
yeast	Hamming Loss	0.215	**0.206**	0.009
	Accuracy	0.400	**0.493**	-0.093
	Precision	**0.724**	0.693	0.031
	Recall	0.471	**0.580**	-0.109
	F_1	0.570	**0.632**	-0.062
scene	Hamming Loss	0.113	**0.098**	0.015
	Accuracy	0.502	**0.610**	-0.108
	Precision	**0.707**	0.630	0.077
	Recall	**0.634**	0.580	0.054
	F_1	**0.668**	0.632	0.036
corel5k	Hamming Loss	0.010	**0.009**	0.001
	Accuracy	0.012	**0.060**	-0.048
	Precision	**0.320**	0.175	0.145
	Recall	0.012	**0.063**	-0.051
	F_1	0.023	**0.093**	-0.070
enron	Hamming Loss	0.058	**0.049**	0.009
	Accuracy	0.146	**0.404**	-0.258
	Precision	**0.688**	0.640	0.048
	Recall	0.157	**0.461**	-0.304
	F_1	0.255	**0.536**	-0.281
medical	Hamming Loss	0.012	**0.011**	0.001
	Accuracy	0.631	**0.713**	-0.082
	Precision	**0.780**	0.760	0.020
	Recall	**0.767**	0.740	0.027
	F_1	**0.774**	0.750	0.024

The proposed method IRL-MLC obtained better results than OSML-ELM method in nine of twenty five performance

metrics. It should also be noted that the proposed method presents, in addition to what was possible to conclude based on numerical metrics, an advantage over the method considered in the comparisons, which is to allow the understanding of the generated rules. The results show that the approach used in this article is promising, since the proposed method presents a performance similar to that of the method considered in the comparisons, which is a state-of -the-art method. The results also suggest that the continuity of research in the direction adopted here will be beneficial for the multi-label classification area, an area that presents many challenges and open problems, which motivate investments in research.

V. Conclusions

This paper detailed a new method for genetic learning of fuzzy rule bases for multi-label classification called IRL-MLC, one of the few algorithm adaptation method based on fuzzy rules and EFS in the state-of-art. The proposed method is divided into two steps. The first step is based on an iterative genetic learning, where a GA is run a number of times, for each time adding the best fuzzy rule to a preliminary RB; this step shows a novel if-then fuzzy rule fitness for multi-label classification problem based on the rule degree of firing with each example in the dataset. The second step is based on GA to tune the preliminary RB and defines the fuzzy rule weights and threshold value; this step shows a novel way to assign a set of labels to each example. The final RB is formed by the all fuzzy rules in the preliminary RB, all fuzzy rule weights and threshold value. Finally, the results show that the proposed method is competitive with another state-of-art method.

Future research work directions include: to use other types of evolutionary algorithms like multi objective evolutionary algorithms (NSGA-II, SPEA2, PESA-II, MOEA/D, among others) to considerer an interpretability objective with some interpretability indexes; to use membership function tuning to allow MFs to present the real distribution of data; to use the proposed method on other types of datasets; and, to compare the proposed method with other state-of-art methods.

Acknowledgment

This work was supported by the Universidad Nacional de San Agustín de Arequipa under Project IBAIB-06-2019-UNSA.

References

[1] S. Yang, S. Kim, and Y. Man Ro, "Semantic home photo categorization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 3, pp. 324–335, March 2007.

[2] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Multilabel text classification for automated tag suggestion," 2008.

[3] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang, "Correlative multi-label video annotation," pp. 17–26, 2007.

[4] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. P. Vlahavas, "Multi-label classification of music into emotions." pp. 325–330, 2008.

[5] M. Nareshpalsingh and N. Modi, "Multi-label classification methods: A comparative study," vol. 4, no. 12, pp. 263–270, 2017.

[6] M. Majzoubi and A. Choromanska, "Ldsm: Logarithm-depth streaming multi-label decision trees," *CoRR*, vol. abs/1905.10428, 2019.

[7] P. Skryjomski, B. Krawczyk, and A. Cano, "Speeding up k-nearest neighbors classifier for large-scale multi-label learning on gpus," *Neurocomputing*, vol. 354, pp. 10 – 19, 2019.

[8] S. Abe, "Fuzzy support vector machines for multilabel classification," *Pattern Recogn.*, vol. 48, no. 6, pp. 2110–2117, Jun. 2015.

[9] J. Wang, J. Feng, X. Sun, S.-S. Chen, and B. Chen, "Simplified constraints rank-svm for multi-label classification," *Communications in Computer and Information Science*, vol. 483, pp. 229–236, 2014.

[10] B. Al-Salemi, M. J. A. Aziz, and S. A. Noah, "Boosting algorithms with topic modeling for multi-label text categorization: A comparative empirical study," *Journal of Information Science*, vol. 41, no. 5, pp. 732–746, 2015.

[11] J. Aslam Parwez, Muhammad Abulaish, "Multi-label classification of microblogging texts using convolution neural network," *IEEE Access*, vol. 7, no. 99, pp. 68 678–68 691, 2019.

[12] I. Mitiche, A. Nesbitt, P. Boreham, B. G. Stewart, and G. Morison, "Naive bayes multi-label classification approach for high-voltage condition monitoring," pp. 162–166, Nov 2018.

[13] A. Fernández, V. López, M. J. del Jesus, and F. Herrera, "Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges," *Knowledge-Based Systems*, vol. 80, pp. 109 – 121, 2015.

[14] R. C. Prati, "Fuzzy rule classifiers for multi-label classification," *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–8, 2015.

[15] J. C. Hühn and E. Hüllermeier, "Furia: an algorithm for unordered fuzzy rule induction," *Data Mining and Knowledge Discovery*, vol. 19, pp. 293–319, 2009.

[16] V. Bansode and S. S. Sane, "Multi-label classification methods: A comparative study," *International Journal of Advance Research and Innovative Ideas in Education (IJARIIE)*, vol. 3, no. 5, pp. 2395–4396, 2017.

[17] H.-Y. Lo, S. de Lin, and H.-M. Wang, "Generalized k-labelsets ensemble for multi-label and cost-sensitive classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, pp. 1679–1691, 2014.

[18] E. H. Cárdenas and H. A. Camargo, "A multi-objective evolutionary algorithm for tuning type-2 fuzzy sets with rule and condition selection on fuzzy rule-based classification system," pp. 389–399, 2018.

[19] E. C. Hinojosa and H. A. Camargo, "Multi-objective evolutionary algorithm for tuning the type-2 inference engine on classification task," *Soft Comput.*, vol. 22, no. 15, pp. 5021–5031, Aug. 2018.

[20] E. H. Cárdenas, H. A. Camargo, and Y. J. Túpac, "Imbalanced datasets in the generation of fuzzy classification systems - an investigation using a multiobjective evolutionary algorithm based on decomposition," pp. 1445–1452, 2016.

[21] C. E. Hinojosa, H. A. Camargo, and V. Y. J. Tpac, "Learning fuzzy classification rules from imbalanced datasets using multi-objective evolutionary algorithm," pp. 1–6, Oct 2015.

[22] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, Dec 1999.

[23] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," *Data Mining and Knowledge Discovery Handbook*, pp. 667–685, 2010.

[24] M. Pushpa and S. Karpagavalli, "Multi-label classification: Problem transformation methods in tamil phoneme classification," *Procedia Computer Science*, vol. 115, pp. 572 – 579, 2017.

[25] E. A. Cherman, M. C. Monard, and J. Metz, "Multi-label problem transformation methods: a case study," *CLEI Electronic Journal*, vol. 14, no. 1, pp. 4–4, 2011.

[26] J. Montañés, E. Quevedo and del Coz J.J., "Improving stacking approach for multi-label classification," *Proceedings of the 2011 Spanish Conference on Artificial Intelligence*, pp. 484–500, 2011.

[27] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press, 1975.

[28] P. Nordin, "Evolutionary program induction of binary machine code and its applications," Ph.D. dissertation, der Universitat Dortmund am Fachereich Informatik, 1997.

[29] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework." *Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

[30] R. Venkatesan, M. J. Er, M. Dave, M. Pratama, and S. Wu, "A novel online multi-label classifier for high-speed streaming data applications," *Evolving Systems*, vol. 8, no. 4, pp. 303–315, Aug 2016.