

FedCSIS_2023_paper_4139.pdf

Explainability in RIONA Algorithm Combining Rule Induction and Instance-Based Learning

Grzegorz Góra

University of Warsaw
Banacha 2, 02-097 Warszawa
Poland
Email: ggora@mimuw.edu.pl

Andrzej Skowron

Systems Research Institute PAS
Newelska 6, 01-447 Warszawa
Poland
Email: skowron@mimuw.edu.pl

Arkadiusz Wojna

DeepSeas
12121 Scripps Summit Court
San Diego, CA 92131, USA
Email: wojna@mimuw.edu.pl

Abstract—The article concerns the well-known RIONA algorithm. The focus on the explainability property of this algorithm. The theoretical results, formulated and proved in the paper, show the relationships of the RIONA classifiers to both instance- and rule-based classifiers. In particular, we show the equivalence (relative to the classification) of the RIONA algorithm with the rule-based algorithm generating all consistent and maximally general rules from the neighbourhood of the test case.

I. INTRODUCTION

In the paper, we focus on the learning algorithm for supervised learning [28], [19], [21]. Specifically, we focus on the well-known RIONA algorithm [14], [13], [12]. This algorithm combines two widely-used empirical approaches: rule induction and instance-based learning. Both these approaches use reasoning schemes easily understandable by a human. It is important since, in the last years, one can observe rapidly increasing importance of this issue in real-life applications. This is related to the topic of so-called explainable Artificial Intelligence (see e.g. [2], [10], [18]). Together with the decision for the given test object the classifying system should provide an explanation for this decision understandable by the user. In the paper, we present theoretical results of considered algorithm that allow to meet these requirements.

The RIONA algorithm is based on a few ideas. First, it computes rules in a lazy manner (see e.g. [3]), that is it induces a very limited set of decision rules relevant only for the test example. This is a different strategy than inducing a large number of decision rules in advance to use them during testing.

Second, the classification performed by RIONA on a given test object is based on rules induced only from the neighbourhood of the given test example. Note that a small number of rules is sufficient when the lazy approach is applied.

Third, we use a different kind of rules than those commonly used in rule-based approaches, where conditions are of the form: *attribute equal to the specific value*. In RIONA, more general rules are used with conditions of the form: *attribute belongs to a set of values*. These sets of values are specified by grouping both numerical and symbolic values of attributes. In voting for the decision by rules covering the example being classified, the aggregation of the support sets of such rules is used.

Fourth, RIONA constructs object neighbourhoods of the optimal size.

The properties of RIONA algorithm are worth studying as it was reported in the literature as one of the most accurate classification methods in many experimental comparisons done by various researchers (the most commonly used RIONA implementation is a classifier in the WEKA platform named RseplibKnn [1]), to name a few: Facebook content recognition [9] Chapter 1] (RIONA was the best one of 21 tested algorithms), environmental sound recognition [15] (best of 9 algorithms), metabolic pathway prediction of plant enzymes [8] (2nd of 47 algorithms), acoustic-based environment monitoring [29] (2nd of 8 algorithms), context awareness of a service robot [16] (2nd of 8 algorithms), student performance prediction [4] (5th of 47 algorithms).

In the paper, we present the theoretical results, which show the relationships of the RIONA classifiers to both instance- and rule-based classifiers. In particular, we show the equivalence (relative to the classification) of the RIONA algorithm with the rule-based algorithm generating all consistent and maximally general rules from the neighbourhood of the test case. Consequently, the RIONA classifier can be represented by a rule-based classifier, with rules easily interpretable by humans. These theoretical results provide the explainability of the resulting classifiers of RIONA and could be used in the situation when an explanation or justification of the derived decision is important.

II. BASIC NOTIONS

The domain of learning is a space of objects X . Each object $x \in X$ is described by a finite set of pairs $(a, a(x))$, where a is a conditional attribute from a given set A of (conditional) attributes, i.e. $a : X \rightarrow V_a$ for $a \in A$, where the codomain V_a of a is the set of values of a and $a(x)$ is the value of a on the object $x \in X$. We consider two types of attributes: numerical and symbolic. We denote the sets of symbolic and numerical attributes respectively by A_{sym} and A_{num} .

Any triplet (X, A, d) , where X is the space of objects, A is a set of attributes and d is a decision function, is called *decision system*. Let (X, A, d) be a decision system and for any attribute $a \in A$, ϱ_a be a pseudometric on the respective value set V_a , i.e. for any $a \in A$, (V_a, ϱ_a) is a pseudometric space.

We call such an enriched decision system the *pseudometric decision system* and denote it by $(\mathbf{X}, A, d, \{\varrho_a\}_{a \in A})$.

A. Rule-based methods

One of the critical techniques is the induction of rule sets (see e.g. [11]). Its importance follows from the fact that knowledge representation in the form of rules is well understandable by a human. We are interested in decision rules which indicate what decision should be taken in a perceived situation. In the most general form, *decision rules* are of the form ‘if φ then ψ ’, where φ is called the premise of the rule and ψ its consequence. ψ is a formula determined by a decision attribute d .

Rule induction algorithms induce decision rules from a training set. We define what kind of rules we admit and in consequence what kind of rules we search for. We consider decision rules with premises consisting of a conjunction of *elementary conditions* and their consequences indicating the specific decision. Each elementary condition describes a set of values of the attribute. Informally, it is of the form $a \in V$, where $V \subseteq V_a$. First, we define how such sets V of values can be expressed over a formal language together with semantics (meaning) of expressions from this language in the power set of attribute codomain.

Definition II.1. Let $\mathbb{D} = (\mathbf{X}, A, d)$ be a decision system (or $\mathbb{D} = (\mathbf{X}, A, d, \{\varrho_a\}_{a \in A})$ be a pseudometric decision system). The description of any elementary set for symbolic attributes $a \in A_{sym}$ is one of the following forms:

$$\emptyset \quad (1)$$

$$\{v\}, \text{ where } v \in V_a, \quad (2)$$

$$1, \quad (3)$$

$$B(c, r), \text{ when } \mathbb{D} \text{ is pseudometric decision system and } \quad (4)$$

$$\text{where } c \in V_a, r \in \mathbb{R}, r \geq 0.$$

The description of elementary set for decision attribute d is of the form [2] where attribute a is substituted by d .

The description of elementary set for numerical attributes $a \in A_{num}$ is of the following form:

$$\emptyset \quad (5)$$

$$[b, e], (b, e], [b, e), (b, e), \text{ where } b, e \in \mathbb{R} \text{ are such that} \quad (6)$$

$$\text{the corresponding interval between points } b \text{ and } e \text{ is included in } V_a.$$

The semantics $\|des\|_{\mathbb{D}}$ of any description des of the elementary set for attribute $a \in A \cup \{d\}$ is defined as a subset of V_a as follows¹:

$$\|\emptyset\|_{\mathbb{D}} = \emptyset,$$

$$\|\{v\}\|_{\mathbb{D}} = \{v\} \text{ (it is called the singleton set),}$$

$$\|V_a\|_{\mathbb{D}} = V_a \text{ (it is called the value set of } a),$$

$$\|[b, e]\|_{\mathbb{D}} = [b, e], \|(b, e)\|_{\mathbb{D}} = (b, e),$$

¹For simplicity we do not distinguish between symbols denoting values and values themselves.

$$\|[b, e]\|_{\mathbb{D}} = [b, e], \|(b, e)\|_{\mathbb{D}} = (b, e),$$

$$\|B(c, r)\|_{\mathbb{D}} = \{w \in V_a : w \in B(c, r)\}$$

$$= \{w \in V_a : \varrho_a(c, w) \leq r\} \text{ (called the ball set).}$$

Now, we define the representation of the elementary conditions (in a language) and their semantics.

Definition II.2. Let $\mathbb{D} = (\mathbf{X}, A, d)$ be a decision system (or $\mathbb{D} = (\mathbf{X}, A, d, \{\varrho_a\}_{a \in A})$ be a pseudometric decision system).

The elementary condition for attribute $a \in A \cup \{d\}$ has the form: $a \in V$, where $a \in A$ and V is a description of elementary set for attribute a . Its semantics is defined as follows: $\|[a \in V]\|_{\mathbb{D}} = \{x \in \mathbf{X} : a(x) \in \|V\|_{\mathbb{D}}\}$

The semantics of $a \in V$ may be restricted to subsets of $\mathbf{X}$, e.g. to the training set, $trnSet$, i.e. $\|[a \in V]\|_{\mathbb{D}} \cap trnSet$ denoted as $\|[a \in V]\|_{trnSet}$. The elementary condition t is satisfied by an example x (or, in short, $t(x)$ is satisfied) if $x \in \|[t]\|_{\mathbb{D}}$.

One should note that $\|V\|_{\mathbb{D}}$ denotes a subset of the attribute value set of a given attribute, while $\|[a \in V]\|_{\mathbb{D}}$ denotes a subset of $\mathbf{X}$. Each elementary condition is of the form: $a \in V$, where $\|V\|_{\mathbb{D}} \subseteq V_a$ and $\|V\|_{\mathbb{D}}$ is

- a singleton set for decision attribute (see set description [4] and its semantics),
- a proper interval for the numerical attribute (see set description [6] and its semantics), and
- a singleton set, value set of an attribute or a ball set for the symbolic attribute (see set descriptions [2] [3] [4] respectively and their semantics).

The elementary condition is satisfied for a given object if the value of the concerned attribute on this object belongs to the set given by its description. Conditions of the form $a \in \{v\}$, where $v \in V_a$ are also written as $a = v$. Conditions of the form $a \in V_a$ which are always true (i.e. the set of objects satisfying the condition is equal to the set of all objects in the considered universe), also written as $a = *$, are called *trivial*.

Finally, we define the semantics and the syntax for expressing premise and consequence of the decision rules.

Definition II.3. Let $\mathbb{D} = (\mathbf{X}, A, d)$ be a decision system (or $\mathbb{D} = (\mathbf{X}, A, d, \{\varrho_a\}_{a \in A})$ be a pseudometric decision system). A decision rule is an expression of the form

$$\text{if } t_1 \wedge t_2 \wedge \dots \wedge t_m \text{ then } d = v,$$

where m is the number of attributes, t_i is an elementary condition for an attribute a_i for $i = 1, 2, \dots, m$, $v \in V_d$.

The semantics of the premise of the rule r , φ is defined as follows:

$$\|[\varphi]\|_{\mathbb{D}} = \|[t_1 \wedge t_2 \wedge \dots \wedge t_m]\|_{\mathbb{D}} = \|[t_1]\|_{\mathbb{D}} \cap \|[t_2]\|_{\mathbb{D}} \dots \cap \|[t_m]\|_{\mathbb{D}}$$

The premise of the rule r , φ , is satisfied by example x (or x satisfies φ) if $x \in \|[\varphi]\|_{\mathbb{D}}$. In this case, example x is said to match the rule r and r is said to cover x .

The single rule is a classifier which classifies examples covered by that rule to the decision class indicated by the rule's

consequence. Ideally, we could search for rules *if* φ *then* $d = v$ such that $[[\varphi]]_{\mathbb{D}} \subseteq [[d = v]]_{\mathbb{D}}$. However, the semantics of $[[d = v]]_{\text{trnSet}}$ is available only. Hence, we induce rules for trnSet and assume that the inclusion extends on $\mathbf{X}$. Moreover, we search for rules covering as many as possible examples.

Usually, while presenting the decision rule, trivial conditions are omitted². The commonly used conditions for symbolic attributes are equations $a = v$, while for numerical attributes conditions are specified by interval inclusions, e.g.:

if $a_1 = 2 \wedge a_3 \in [3, 7] \wedge a_6 = 5$ *then* $d = 1$.

However, for symbolic attributes we use a more general condition such as $a \in V$ (see Definition II.2), which is introduced to extend the notion of the single sets to the ball sets specified by form 4 in Definition II.1 and its semantics. If the data set of the considered problem contains some numerical attributes, then the relevant intervals can be obtained by applying discretisation. Discretisation transforms decision system into a new one in such a way that numerical values are grouped into relevant intervals covering the whole attribute domain. Consecutive intervals induced from the original table are mapped into successive numbers representing values of the discretised attribute in a new decision system (see e.g. [25]).

For any decision rule¹, we denote by $t_i(r)$ the i -th condition t_i from Definition II.3 for rule r ; we denote by $t_a(r)$ for $a \in A$ the condition t_i from Definition II.3 for rule r corresponding to attribute a .

In the paper, we consider three kinds of decision rules relative to the admissible elementary conditions used in Definition II.3. Below we specify three possibilities¹ admissible elementary conditions used in Definition II.3. They specify three kinds of decision rules, and in consequence, three sets of decision rules¹.

Definition II.4. Let $(\mathbf{X}, A, d)$ be a decision system.

For the data sets with only symbolic attributes the set of simple rules denoted as SimRules is the set of all rules from Definition II.3 in which the only admissible elementary conditions in the premise of the rule are from set descriptions 2 3 i.e. elementary conditions are $a = v$ for $v \in V_a$; and $a = *$.

The set of combined rules denoted as CombRules is the set of all rules from Definition II.3 for which the only admissible condition¹ in the premise of the rule are from set descriptions 2 3 6 i.e. elementary condition for symbolic attributes are as in the definition of SimRules and for numerical attributes are of the form $a \in I$, where I is a proper interval description.

Definition II.5. Let $(\mathbf{X}, A, d, \{\varrho_a\}_{a \in A})$ be a pseudometric decision system.

Suppose that for all symbolic attributes $a \in A_{\text{sym}}$ there is given a specific value $c_a \in V_a$. The set of general rules denoted as GenRules ($\{(\varrho_a, c_a)\}_{a \in A_{\text{sym}}}$) or simply GenRules (whenever pairs (ϱ_a, c_a) are clear from the context or irrelevant due to generality) is the set of all rules from Definition II.3¹

²In fact, in the description of rules only non-trivial conditions are used. We use trivial conditions only to make the notation simpler.

¹for which the only admissible elementary conditions in the premise of the rule contain set descriptions (i) as in the definition of CombRules for numerical attributes and (ii) specific form of 4 i.e. $B(c, r)$, where $c = c_a$, $r = \varrho_a(c_a, v)$, $v \in V_a$ for symbolic attributes $a \in A_{\text{sym}}$.

The definition of general rules will become more clear after reading Section III.A where it is used.

Definition II.6. A rule r with the consequent ($d = v$) is consistent with a set of objects $X \subseteq \mathbf{X}$ (sometimes we write simply consistent whenever set X is clear from the context) if for each object $x \in X$ whenever x matches the rule r the decision of the rule is identical with the decision of the object, i.e. $d(x) = v$.

A rule r is inconsistent if it is not consistent.

Usually, in the above definition we use as a set X , the set of training objects. A rule consistent with the training set classifies correctly all the training examples covered by that rule.

Now we define the notion of maximality of rule.

Definition II.7. Let r_1 and r_2 be rules. We say that a condition $t_i(r_2)$ is more general than (or is implied by) a condition $t_i(r_1)$, in symbols $t_i(r_1) \Rightarrow t_i(r_2)$, if $||V_1||_{\mathbb{D}} \subseteq ||V_2||_{\mathbb{D}}$ holds, where $t_i(r_1)$ is of the form $a_i \in V_1$ and $t_i(r_2)$ is of the form $a_i \in V_2$.

We say that a rule r_2 is more general than (or is implied by) a rule r_1 , and denote it by $r_1 \Rightarrow r_2$ if it has identical consequents, i.e. $d(r_1) = d(r_2)$ and each condition $t_i(r_2)$ is more general than condition $t_i(r_1)$ (for $i = 1, 2, \dots, m$).

A consistent rule r with a training set trnSet is maximally general (relative to this training set and a given set of rules) if there is no rule in this set of rules more general than r which is different from r and consistent with trnSet .

Definition II.8. For a given set of admissible rules Rules , a training set trnSet we define the set of maximally general rules $\text{MaxRules}(\text{Rules}, \text{trnSet})$ to be equal to the set of all rules $r \in \text{Rules}$ consistent with trnSet and maximally general.

In the paper, we consider only sets of rules Rules in Definition II.8 equal to one of three sets: SimRules , CombRules , GenRules from Definitions II.4 and II.5. If the sets Rules and trnSet are obvious from the context, we write MaxRules instead of $\text{MaxRules}(\text{Rules}, \text{trnSet})$. Also, we write MaxRules in general case, i.e. MaxRules denotes any one of the three mentioned cases: $\text{MaxRules}(\text{SimRules}, \text{trnSet})$, $\text{MaxRules}(\text{CombRules}, \text{trnSet})$, $\text{MaxRules}(\text{GenRules}, \text{trnSet})$.

From the knowledge discovery perspective, the important problem is to compute all rules (matched at least by one training example) that are maximally general and consistent with a training set.

Let us first consider $\text{MaxRules}(\text{SimRules}, \text{trnSet})$. In this case, a consistent rule is maximally general in a training

set $trnSet$ if for each non-trivial condition replacement of that condition with trivial condition makes the rule inconsistent with the training set $trnSet$. Hence, maximally general rules are those which have minimal lengths, where the length of the rule is the number of non-trivial conditions in it. Thus the problem here is to find the complete set of consistent and minimal decision rules (see e.g. [31]). Among different aspects, such a set of rules is also essential because it relates to the minimal description length principle (see e.g. [30]). Algorithms for computing all minimal rules are very time consuming, especially when the number of training objects or attributes is significantly large. This is because the size of the $MaxRules$ set can be exponential concerning the size of the training set (see e.g. [22]). In practice, approximation algorithms are often applied to obtain the rule set that is not necessarily complete (see e.g. [7]). There are also other approaches to induce a set of rules, which cover the input examples using, e.g. the smallest number of rules (see e.g. [17]). However, in the paper, we focus on the complete $MaxRules$ set.

Now, let us consider $MaxRules(CombRules, trnSet)$. In this case, additionally we have numerical attributes for which maximally general intervals are searched. Searching for maximally general rules for numerical attributes relates to the problem of discretisation. A partition of discretisation is consistent if each interval covers only objects with the same decision. For more details on discretisation, the readers are referred to [25], [23].

It should be noted that the problem of searching for a consistent partition with the minimal number of cuts is NP-hard (see e.g. [25]). It shows that the problem of discretisation from the global point of view is a complex task. We will show in Subsection III-A that it is in a sense possible to overcome this problem if one focuses on a local fragment of the universe instead of the whole universe. It occurs in case of the presented lazy rule induction algorithm (see Algorithm 3 or Algorithm 4).

Now, let us consider $MaxRules(GenRules, trnSet)$. In this case, we additionally search for relevant grouping of values for symbolic attributes. It relates to the problem of partition of symbolic attributes. Formally the partition over an attribute a is any function $P_a : V_a \rightarrow \{1, \dots, m_a\}$. The problem of searching for a consistent family of partitions with the minimal $\sum_{a \in A} |P_a(V_a)|$ is NP-hard (see e.g. [24]). We overcome this because of two reasons. First, we limit the number of possible groupings of values of any attribute (from 2^n to n^2 , where n is the number of values for an attribute). Second, we use lazy rule induction (see Section III-A).

Rules induced from training examples are then used to classify objects. For a given test object, the subset of rules matched by the object is selected. If the object matches only rules with the same decision, then the decision predicted by those rules is assigned to the example. If the test object matches the rules corresponding to different decisions, the conflict has to be resolved (see e.g. [20]). A common approach is to use a measure for conflict resolution, and decision with the highest value of the measure is selected. In the paper, we

focus on the commonly used measure that is presented below.

Definition II.9. Suppose training set $trnSet$, test example tst and $MaxRules$ are given. Then we define

$$Strength(tst, v) = \left| \bigcup_{r \in MatchR(tst, v)} supportSet(r) \right|, \quad (7)$$

where v denotes the v -th decision ($v = 1, \dots, n_d$), $supportSet(r)$ is a set of training examples matching the rule r , $MatchR(tst, v)$ is a subset of $MaxRules$, whose premise is satisfied by tst and the consequent is a decision v , $|\cdot|$ denotes cardinality (size) of a set.

The measure $Strength$ counts the number of training examples covered by the maximally general rules with the decision v and covering a test example tst .

The classifier based on maximally general rules with the measure $Strength$ as a strategy for conflict resolution predicts the decision that is the most frequent in the set of training examples covered by rules matched by a test example, i.e.:

$$decision_{MaxRules}(tst) = \arg \max_{v \in V_d} Strength(tst, v). \quad (8)$$

As mentioned previously, the limitation of this approach lies in the fact that computing $MaxRules$ is very time-consuming.

B. Lazy rule learning for symbolic attributes

Another approach can be based on a construction of algorithms that do not require calculating the set of decision rules before classifying new objects. These are *lazy learning* (or *memory based learning*) algorithms.

The most common example of such algorithm is kNN (see Algorithm 1). In this algorithm and later, we consider the neighbourhood $N(tst, trnSet, k, \rho)$ (N for short) defined as the set of k training examples $trnSet$ that are most similar to tst according to the distance function ρ . In the case when more than one example has the same distance from the object tst to the k -th nearest example, all of them are added to $N(tst, trnSet, k, \rho)$ (then the set $N(tst, trnSet, k, \rho)$ contains more than k examples).

Another, important for our considerations example of lazy rule-based learning algorithm for the case of *SimRules* is presented in [5]. It generates only decision rules relevant for a new test object and then classifies it like algorithms generating rules in advance. It uses a technique that computes the measures from Eqn. 7 for every test object without computing all maximally general rules ($MaxRules$).

First, we define *simple local decision rule*, denoted by $s-rule(tst, trn)$, where tst, trn are the distinguished objects. This name corresponds to the set of simple rules, denoted by $SimRules$ (in the following proposition we show their actual relationship).

³Such solution is used in the proposed RIONA algorithms in the paper and its implementation. Thus we also use it for the kNN algorithm.

Algorithm 1: $kNN(tst, trnSet, k, \varrho)$

Input: a test example tst , training set $trnSet$, positive integer k , pseudometric ϱ

Output: predicted decision for tst

```

1 begin
2   neighbourSet =  $N(tst, trnSet, k, \varrho)$ 
3   foreach decision  $v \in V_d$  do
4     supportSet( $v$ ) =  $\emptyset$ 
5   end
6   foreach  $trn \in neighbourSet$  do
7      $v = d(trn)$ 
8     supportSet( $v$ ) = supportSet( $v$ )  $\cup \{trn\}$ 
9   end
10  return  $\arg \max_{v \in V_d} |supportSet(v)|$ 
11 end

```

Definition II.10. For any test object tst and any training object trn , we define a simple local decision rule (for short s-rule), denoted by $s\text{-rule}(tst, trn)$, the decision rule with the decision $d(trn)$ and the following conditions t_a for each symbolic attribute a :

$$t_a = \begin{cases} a = a(trn) & \text{if } a(tst) = a(trn) \\ a = * & \text{if } a(tst) \neq a(trn) \end{cases}$$

Let us recall that $a = *$ denotes the trivial condition $a \in V_a$. A local decision rule is defined to ensure that both trn and tst objects satisfy the rule and it is maximally specific (the number of trivial conditions is minimal; or inversely, the number of non-trivial conditions is maximal). We have the following crucial relation between s-rule and maximally general consistent rules from *SimRules*:

Theorem II.1. [5]⁴ The rule $s\text{-rule}(tst, trn)$ for a test object tst and a training object trn is consistent with the training set $trnSet$ if and only if there exists a rule in the set $MaxRules(SimRules, trnSet)$ covering objects tst and trn .

It means that for $MaxRules(SimRules, trnSet)$ for any test example tst , any decision $v \in V_d$ computing the value of measure $Strength(tst, v)$ from Eqn. 7 is equivalent to computing the number of training examples $trn \in trnSet$ such that $d(trn) = v$ and rule $s\text{-rule}(tst, trn)$ is consistent with $trnSet$. This is realised by the simple lazy rule induction algorithm for symbolic attributes (LAZY) presented in Algorithm 3.

The function $isCons(r, verifySet)$ checks if a decision rule r is consistent with a $verifySet$. For every training object trn , Algorithm 3 constructs the rule $s\text{-rule}(tst, trn)$ based on the examples tst and trn . Then it checks whether

⁴The original formulation of this proposition was different. However, this formulation in the considered case of *SimRules* is equivalent to the original proposition. Such formulation allows us to show a more direct relationship between local rules and *MaxRules* and algorithms based on these two types of rules.

Algorithm 2: $isCons(r, verifySet)$

Input: a rule $r : \text{if } \alpha \text{ then } d = v$, set of examples $verifySet$

Output: true if rule r is consistent with $verifySet$, false otherwise

```

1 for all  $trn \in verifySet$  do
2   if  $d(trn) \neq v$  and  $trn$  satisfies  $\alpha$  then
3     return false
4   end if
5 end for
6 return true

```

Algorithm 3: $LAZY(tst, trnSet)$

Input: test example tst , training set $trnSet$

```

1 for all decision  $v \in V_d$  do
2   supportSet( $v$ ) =  $\emptyset$ 
3 end for
4 for all  $trn \in trnSet$  do
5    $v = d(trn)$ 
6   if  $isCons(s\text{-rule}(tst, trn), trnSet)$  then
7     supportSet( $v$ ) = supportSet( $v$ )  $\cup \{trn\}$ 
8   end if
9 end for
10 return  $\arg \max_{v \in V_d} |supportSet(v)|$ 

```

the rule $s\text{-rule}(tst, trn)$ is consistent with the remaining training examples, i.e. if all the training examples satisfying the left-hand side of $s\text{-rule}(tst, trn)$ are labelled by the same decision as the considered training example trn . If the rule $s\text{-rule}(tst, trn)$ is consistent, then the training example trn is added to the support set of the relevant decision. Finally, the algorithm selects the decision with the support set of the highest cardinality. As it was mentioned above from Theorem II.1 we have:

Corollary II.2. Let $trnSet$ be a training set. For any test object tst , the classifier from Eqn. 8 with $MaxRules = MaxRules(SimRules, trnSet)$, we have $LAZY(tst, trnSet) = decision_{MaxRules}(tst)$.

Comparison of the LAZY algorithm to the algorithm based on maximally general rules allows us to conclude that the LAZY algorithm considers only the decision rules that can be involved in the classification of a given test object.

III. RIONA DESCRIPTION

A. Extension and generalisation of lazy rule learning

In this section, we present an extension and generalisation of the LAZY algorithm (see Algorithm 3) presented in Subsection II-B. We extend this algorithm to the case of numerical attributes and we use more general conditions for symbolic attributes.

We present our final idea, in three steps. The first step is described in Subsection II-B. In two remaining steps,

presented in this section, we use a generalisation of rules from the previous step (see Subsections III-A1 III-A2).

Thus, in this section together with the first step from Subsection II-B we define three types of local rules: simple local decision rule, combined local decision rule and generalised local decision rule (for short, s-rule, c-rule, and g-rule, respectively), denoted by $s\text{-rule}(tst, trn)$, $c\text{-rule}(tst, trn)$, $g\text{-rule}(tst, trn, \{\varrho_a\}_{a \in A_{sym}})$ (or simply $g\text{-rule}(tst, trn)$), respectively, where trn is the training object, tst is the test object and ϱ_a for $a \in A_{sym}$ are pseudometrics defined by pseudometric decision system. The introduced names correspond to the sets composed out of simple rules, combined rules and general rules, denoted by $SimRules$, $CombRules$, $GenRules$, respectively (see Subsection II-A). In Subsection II-B an important relation between any s-rule and the set of maximally general consistent rules $MaxRules(SimRules, trnSet)$ is presented. In this section, we show analogous important relations between any c-rule or g-rule with their corresponding sets of maximally general consistent rules (denoted by $MaxRules(CombRules, trnSet)$ and $MaxRules(GenRules, trnSet)$, respectively).

1) *Extension of lazy rule learning for numerical attributes:* Throughout this section, we assume that a decision system $\mathbb{D} = (\mathbf{X}, A, d)$ and a training set $trnSet \subseteq \mathbf{X}$ are given.

In the second step, we define an extension of the local decision rule to the case of both symbolic and numerical attributes.

Definition III.1. For any test object tst and any training object trn , we define the combined local decision rule (for short c-rule), denoted by $c\text{-rule}(tst, trn)$, with the decision $d(trn)$ and the following conditions T_a for each attribute $a \in A$:

$$T_a = \begin{cases} a \in [\min_a, \max_a] & \text{if } a \text{ is numerical} \\ t_a & \text{if } a \text{ is symbolic} \end{cases}$$

where t_a is defined as in Definition II.10 $\min_a = \min(a(tst), a(trn))$, $\max_a = \max(a(tst), a(trn))$.

For numerical attributes (linearly ordered conditions are represented in the form $a \in [\min_a, \max_a]$). The interval's endpoints are determined by the attribute values of the examples tst and trn used to form the rule.

It should be noted that by defining c-rule in such a way we obtain an analogous relationship of the set $MaxRules(CombRules, trnSet)$ and $c\text{-rule}(tst, trn)$ to the relation between $MaxRules(SimRules, trnSet)$ and $s\text{-rule}(tst, trn)$.

Lemma III.1. Any rule

$r \in MaxRules(CombRules, trnSet)$ covering the given test object tst and training object trn is implied by the rule $c\text{-rule}(tst, trn)$.

Proof. Since r covers tst and trn and is consistent (with $trnSet$), we have the following. For each attribute $a \in A$, $t_a(r)(trn)$ is satisfied and $t_a(r)(tst)$ is satisfied, i.e. $trn \in [[t_a(r)]]_{\mathbb{D}}$ and $tst \in [[t_a(r)]]_{\mathbb{D}}$.

For rule r such that the elementary condition $t_a(r)$ is of the form $a \in V$ let us define $V_a(r) = ||V||_{\mathbb{D}}$. We will show that for all $a \in A$ the implication $t_a(c\text{-rule}(tst, trn)) \Rightarrow t_a(r)$ holds, i.e. $V_a(c\text{-rule}(tst, trn)) \subseteq V_a(r)$.

First, let us consider the case when a is symbolic. If $t_a(r)$ is a trivial condition, i.e. $t_a(r)$ is of the form $a \in V_a$, then the implication obviously holds (trivial condition is implied by any condition, because for any elementary condition $a \in V$ for attribute a , $||V||_{\mathbb{D}} \subseteq ||V_a||_{\mathbb{D}}$). Let us consider the case when $t_a(r)$ is of the form $a = v$ for some $v \in V_a$. Then, because $t_a(r)(trn)$ and $t_a(r)(tst)$ are satisfied, then $trn \in [[a = v]]_{\mathbb{D}}$ and $tst \in [[a = v]]_{\mathbb{D}}$, i.e. $a(trn) \in \{v\}$ and $a(tst) \in \{v\}$, thus $v = a(trn) = a(tst)$. It means that $t_a(r) = t_a(c\text{-rule}(tst, trn))$ (see Definition III.1 and Definition II.10).

Second, let us consider the case when a is numerical. Thus $t_a(r)$ is of the form $a \in I$, where I is the interval corresponding to the numerical attribute a of rule r . Because $t_a(r)(trn)$ is satisfied, i.e. $trn \in [[t_a(r)]]_{\mathbb{D}}$ and $t_a(r)(tst)$ is satisfied, i.e. $tst \in [[t_a(r)]]_{\mathbb{D}}$ and by definition $[[a \in I]]_{\mathbb{D}} = \{x \in \mathbf{X} : a(x) \in ||I||_{\mathbb{D}}\}$ we have $a(trn) \in ||I||_{\mathbb{D}}$ and $a(tst) \in ||I||_{\mathbb{D}}$, thus $\{a(trn), a(tst)\} \subseteq ||I||_{\mathbb{D}}$. Thus, all points between $a(trn)$ and $a(tst)$ are also in $||I||_{\mathbb{D}}$. In consequence, $[\min_a, \max_a] \subseteq ||I||_{\mathbb{D}}$, where $\min_a = \min(a(tst), a(trn))$, $\max_a = \max(a(tst), a(trn))$. Thus, $V_a(c\text{-rule}(tst, trn)) \subseteq V_a(r)$ (see Definition III.1). $\square$

Theorem III.2. The rule $c\text{-rule}(tst, trn)$ for the test object tst and the training object trn is consistent with the training set $trnSet$ if and only if there exists a rule $r \in MaxRules(CombRules, trnSet)$ covering objects tst and trn .

Proof. First, we show that if the rule $c\text{-rule}(tst, trn)$ is consistent with the training set $trnSet$, it can be extended to a rule belonging to $MaxRules(CombRules, trnSet)$. We define such a rule inductively. Rule $r_0 = c\text{-rule}(tst, trn)$ is in $CombRules$ and is consistent with $trnSet$ by assumption. The induction step is as follows. To define each next rule r_i , for $i = 1, 2, \dots, m$, where m is the number of attributes, we assume that rule r_{i-1} is consistent with $trnSet$ and conditions $t_j(r_{i-1})$ for all $j = 1, 2, \dots, i-1$ are maximally general, i.e. if we replace any condition t_j with a more general t (i.e. $t_j \Rightarrow t$) preserving consistency, then $t_j = t$.

In the i -th induction step we define the condition $t_i(r_i)$ as the maximal generalisation of the condition $t_i(r_{i-1}) = t_i(r_0) = t_i(c\text{-rule}(tst, trn))$ preserving consistency with $trnSet$. All others conditions and the decision of the rule are defined as in the previous induction step, i.e. $t_j(r_i) = t_j(r_{i-1})$ for $j \neq i$; $d(r_i) = d(r_{i-1})$. In other words, in i -th induction step we simply maximally generalise condition for attribute a_i .

First, let us consider the case when a_i is symbolic. If $t_i(r_{i-1})$ is the trivial condition, then we define $r_i = r_{i-1}$. If $t_i(r_{i-1})$ is non-trivial, we replace it with the trivial condition if such replacement keeps the consistency of the rule; otherwise, we keep $r_i = r_{i-1}$.

Now, let us consider the case when a_i is numerical. Thus $t_i(r_{i-1})$ is of the form $a_i \in [min, max]$. We define by $rule_i(r, t)$ the rule r with the replacement of i -th condition in it by condition t . Let us consider the set of training examples which potentially may violate the consistency of the rule under the maximal possible extension of the condition $t_i(r_{i-1})$, i.e. the set $Inc = \{trn \in trnSet : d(trn) \neq d(r_0) \wedge rule_i(r_{i-1}, a_i = *) \text{ covers } trn\}$. Let us define $a(Inc) = \{a(trn) : trn \in Inc\}$. From the induction assumption, r_{i-1} is consistent with $Inc \subseteq trnSet$. Thus we have $a(Inc) \cap [min, max] = \emptyset$. Let us define $newmax = \min\{v \in a(Inc) : v > max\}$. Let us note that this minimum exists because the Inc set and therefore also $a(Inc)$ are finite sets. If the set $\{v \in a(Inc) : v > max\}$ is empty we define $newmax = u_{a_i}$ (i.e. maximal possible extension of the right end of the interval). Analogously, let us define $newmin = \max\{v \in a(Inc) : v < min\}$. If the set $\{v \in a(Inc) : v < min\}$ is empty we define $newmin = l_{a_i}$ (i.e. maximal possible extension of the left end of the interval). Finally, we define $t_i(r_i)$ as the condition $a \in (newmin, newmax)$. From the definition, r_i is consistent with $trnSet$. It is also maximal because maximally extended ends of the interval (i.e. l_{a_i} or u_{a_i}) cannot be extended and other ends of the interval even if extended by one point to a closed interval will cause inconsistency.

We prove now that all other conditions $t_j(r_i)$ for $j < i$ are still maximal. Let us assume that for some $j < i$ the condition $t_j(r_i)$ could be extended to t with preserving consistency, i.e. $rule_j(r_i, t)$ is consistent. We also have that $rule_j(r_i, t)$ is more general rule than $rule_j(r_{i-1}, t)$ (i -th condition is more general), i.e. $rule_j(r_{i-1}, t) \Rightarrow rule_j(r_i, t)$. Therefore $rule_j(r_{i-1}, t)$ is consistent with $trnSet$. From the induction assumption $t = t_j(r_{i-1})$. Because $t_j(r_{i-1}) = t_j(r_i)$ for ($j < i$), then $t = t_j(r_i)$. It means that $t_j(r_i)$ is maximally general.

By induction, the last rule r_m is consistent with $trnSet$ and maximally general.

Second, we show that if the rule $c\text{-rule}(tst, trn)$ is inconsistent with the training set $trnSet$, then there is no rule in $MaxRules(CombRules, trnSet)$ covering tst and trn . In fact, from Lemma III.1 we have that each rule $r \in MaxRules(CombRules, trnSet)$ covering objects tst and trn is implied by $c\text{-rule}(tst, trn)$. Thus inconsistency of the rule $c\text{-rule}(tst, trn)$ implies inconsistency of all rules $r \in MaxRules(CombRules, trnSet)$ covering tst and trn . $\square$

2) *Generalisation of lazy rule learning for symbolic attributes:* From now on, we assume that a pseudometric decision system $\mathbb{D} = (\mathbf{X}, A, d, \{\varrho_a\}_{a \in A})$ is given. As before, we also assume that a training set $trnSet \subseteq \mathbf{X}$ is given.

In the previous Definitions II.10 and III.1 the trivial condition for symbolic attributes is used (when attribute values of the test and training examples differ). This condition represents the grouping of all possible values of an attribute and is satisfied by any object. However, we noticed that a proper subset of

all attribute values can be more relevant for the classification. Grouping of values can be done using a given pseudometric for the attribute. Hence, as the third and final step, we propose the following generalisation of Definition III.1 which additionally leads to a grouping of values for symbolic attributes:

Definition III.2. For any test object tst and any training object trn , we define the generalised local decision rule (for short *g-rule*), denoted by *g-rule* $(tst, trn, \{\varrho_a\}_{a \in A_{sym}})$ or simply *g-rule* (tst, trn) (if parameters $\{\varrho_a\}_{a \in A_{sym}}$ are clear from the context or irrelevant due to generality of considerations), with the decision $d(trn)$ and the following conditions t_a for each attribute a :

$$t_a = \begin{cases} a \in [min_a, max_a] & \text{if } a \text{ is numerical} \\ a \in B(a(tst), r_a) & \text{if } a \text{ is symbolic,} \end{cases}$$

where $min_a = \min(a(tst), a(trn))$, $max_a = \max(a(tst), a(trn))$, the radius $r_a = \varrho_a(a(tst), a(trn))$, and $B(c, R)$ is a closed pseudometric ball of radius R centred at point c defined by the pseudometric ϱ_a .

Again, it should be noted that for *g-rule* we obtain an analogous relationship of the set $MaxRules(GenRules, trnSet)$ and *g-rule* (tst, trn) to the relation between $MaxRules(SimRules, trnSet)$ and *s-rule* (tst, trn) . This is expressed in the following lemma.

Lemma III.3. Let tst be any test object and trn be any training object. Let $GenRules$ be defined by parameters ϱ_a (from given pseudometric decision system) and $c_a = a(tst)$ for $a \in A_{sym}$ (see Definition II.5), i.e. $GenRules = GenRules(\{(\varrho_a, a(tst))\}_{a \in A_{sym}})$. Then any rule $r \in MaxRules(GenRules, trnSet)$ covering objects tst and trn is implied by the rule *g-rule* $(tst, trn, \{\varrho_a\}_{a \in A_{sym}})$.

Proof. The proof is an extension of proof of Lemma III.1. For numerical attributes, the proof is the same as before.

For symbolic attributes, we substitute the part of the proof of Lemma III.1 by the following one. Let us assume that a is symbolic. Then $t_a(r)$ is of the form $a \in B(a(tst), R_a)$, where $R_a = \varrho_a(a(tst), v)$, for some $v \in V_a$. Hence, because $t_a(r)(trn)$ is satisfied, then $R_a \geq \varrho_a(a(tst), a(trn))$. Thus $B(a(tst), r_a) \subseteq B(a(tst), R_a)$, where $r_a = \varrho_a(a(tst), a(trn))$. Then, $t_a(g\text{-rule}(tst, trn)) \Rightarrow t_a(r)$. $\square$

Theorem III.4. Under the assumptions of Lemma III.3 the rule *g-rule* $(tst, trn, \{\varrho_a\}_{a \in A_{sym}})$ is consistent with the training set $trnSet$ if and only if there exists a rule from $MaxRules(GenRules, trnSet)$ covering the objects tst and trn .

Proof. The proof is a modification of the proof of Theorem III.2.

We substitute the induction step of the proof of Theorem III.2 for the case when a_i is symbolic by the following one. Let us consider the case when a_i is symbolic. Then $t_i(r_{i-1})$ is of the form $a \in B(a(tst), r_a)$, where

$r_a = \varrho_a(a(tst), v)$, for some $v \in V_a$. We consider possible extensions of this condition, i.e. conditions of the form $a \in B(a(tst), R_a)$, where $R_a = \varrho_a(a(tst), w)$, for some $w \in V_a$ such that $R_a \geq r_a$ and the consistency of the rule is preserved (with $trnSet$). Because V_a is finite, then there is a finite number of possible selections, and we choose the one with the maximal value of R_a . The chosen extension is maximally general.

When a_i is numerical, we make the extension of the formula as in Theorem III.2

Analogously as in Theorem III.2 one can conclude that the last rule r_m is consistent with $trnSet$ and maximally general.

Analogously as in Theorem III.2 we obtain (using Lemma III.3) that if the rule $g\text{-rule}(tst, trn)$ is inconsistent with the training set $trnSet$, then there is no rule in $MaxRules(GenRules, trnSet)$ covering tst and trn . $\square$

Let us note that the set $MaxRules(GenRules, trnSet)$ is defined for the given values c_a for $a \in A_{sym}$ (during the testing procedure, for the test example tst , we assume $c_a = a(tst)$). The idea behind the set $MaxRules(SimRules, trnSet)$ was to compute all maximally general rules in advance to use it later during the classification process. In order to compute an analogous set $MaxRules(GenRules, trnSet)$ in advance, this should be done for all possible combinations of all possible values for all symbolic attributes. It would increase the number of generated rules by the factor no more than b^k , where b is the maximal cardinality of $|V_a|$ for $a \in A_{sym}$ and k is the number of symbolic attributes.

Theorem III.4 shows that instead of computing the support sets for rules from $MaxRules(GenRules, trnSet)$ covering a new test case, it is sufficient to generate the g-rules for all training examples and then check their consistency with the training set. This is realised by the lazy Rule Induction Algorithm (RIA) presented below.

The function $isCons(r, verifySet)$ was presented in Subsection II-B in Algorithm 2. The RIA algorithm (see Algorithm 4) differs from the LAZY algorithm (see Algorithm 3) only in line 7 where instead of the rule $s\text{-rule}(tst, trn)$, the rule $g\text{-rule}(tst, trn, \{\varrho_a\}_{a \in A_{sym}})$ is used.

From Theorem III.4 one can conclude that the RIA algorithm computes the measure $Strength$ for $MaxRules = MaxRules(GenRules, trnSet)$. Thus, the results of the mentioned algorithm are equivalent to the results of the algorithm based on calculating $MaxRules$ and using the measure $Strength$ as a strategy for conflict resolution. Hence, we have the corollary analogous to Corollary II.2 (see Subsection II-B).

Corollary III.5. For any test object tst , and the classifier from Eqn. 8 with $MaxRules = MaxRules(GenRules, trnSet)$, where $GenRules = GenRules(\{\{\varrho_a, a(tst)\}\}_{a \in A_{sym}})$, we have

$$RIA(tst, trnSet, \{\varrho_a\}_{a \in A_{sym}}) = decision_{MaxRules}(tst).$$

Algorithm 4: $RIA(tst, trnSet, \{\varrho_a\}_{a \in A_{sym}})$

Input: test example tst , training set $trnSet$, family of pseudometrics for symbolic attributes $\{\varrho_a\}_{a \in A_{sym}}$

Output: predicted decision for tst

```

1 begin
2   foreach decision  $v \in V_d$  do
3      $supportSet(v) = \emptyset$ 
4   end
5   foreach  $trn \in trnSet$  do
6      $v = d(trn)$ 
7     if  $isCons(g\text{-rule}(tst, trn, \{\varrho_a\}_{a \in A_{sym}}), trnSet)$ 
8       then
9          $supportSet(v) = supportSet(v) \cup \{trn\}$ 
10      end
11   end
12   return  $\arg \max_{v \in V_d} |supportSet(v)|$ 
13 end

```

B. Combining instance-based learning and rule methods – RIONA

From now on, we additionally assume that aggregation function Agr is defined as sum of individual metrics unless it is stated differently.

The RIONA algorithm is based on a combination of instance-based learning and rule methods. The primary component of the RIONA algorithm was developed using the observation that the kNN method is widely used, and usually for small values of k , it has quite good performance. On the other hand, in the case of rule-based methods, in general, all training examples are used in the process of rule generation. Based on the observation related to the kNN method, one may suppose that only close training examples to the test case are important in the process of inducing the final decision. In fact, in the RIONA algorithm, the classification of the given test example is based on training objects from a neighbourhood of this example.

Thus, instead of considering all training examples in constructing the support set, like in the RIA algorithm, one can bound it to a certain neighbourhood of a test example. The intuition behind this is that the training examples which are far from a given test object are less relevant for classification than the closer ones.

Now, we are ready to present an approach to inducing of decision that is a kind of combination of instance-based learning and lazy rule learning (see Section III-A). The main idea is based on the strategy for conflict resolution with the use of $Strength$ measure (see Eqn. 7) slightly modified in the

following way:

$$LocStrength(tst, v, k, \varrho) = \left| \bigcup_{r \in MatchR(tst, v)} locSuppSet(r) \right|, \quad (9)$$

where most notation remains the same as in Eqn. 7. $\varrho = Agr(\{\varrho_a\}_{a \in A})$ is the aggregated pseudometric and k is the number indicating the size of the neighbourhood, $locSuppSet(r) = supportSet(r) \cap N(tst, trnSet, k, \varrho)$. The difference lies in the fact that we consider only those examples covered by the rules matched by a test object that are in a specified neighbourhood of the test example. The predicted decision based on the measure *LocStrength* is analogous to the previous one (see Eqn. 8):

$$decLocMaxRules(tst, k, \varrho) = \arg \max_{v \in V_d} LocStrength(tst, v, k, \varrho). \quad (10)$$

In the classification process, we assume that number k for the neighbourhood $N(tst, k)$ is fixed. The proper size of the neighbourhood, i.e. the parameter k is found in the learning phase (see [13]).

Given a set *MaxRules*, the above measures can be calculated by limiting the support sets of the rules covering a test example to the specified neighbourhood of a test example. Thus the algorithm based on maximally general rules with the modified measure *LocStrength* can be used here.

However, the measure *LocStrength* can also be calculated using the lazy rule learning methodology. This is analogous to the fact that the RIA algorithm computes the measure *Strength* (see Corollary III.5). To implement this approach, we modified Algorithm 4. First, in line 5 of the algorithm, only examples $trn \in N(tst, k)$ should be considered. Furthermore, it is not necessary to consider all the examples from the training set to check the consistency of the *g-rule* $(tst, trn, \{\varrho_a\}_{a \in A_{sym}})$ (see line 7 of Algorithm 4). This is due to the following proposition.

Proposition III.6. Suppose that ϱ_a in pseudometric decision system for $a \in A_{num}$ are defined as normalised Euclidean metric. If $trn' \in trnSet$ satisfies *g-rule* $(tst, trn, \{\varrho_a\}_{a \in A_{sym}})$, then $\varrho(tst, trn') \leq \varrho(tst, trn)$, where $\varrho = Agr(\{\varrho_a\}_{a \in A})$ and the aggregation function *Agr* is defined either by sum of metrics or weighted sum of metrics.

Proof. If trn' satisfies *g-rule* $(tst, trn, \{\varrho_a\}_{a \in A_{sym}})$, then from the definition of *g-rule* (see Definition III.2), we have:

- for all symbolic attributes $a \in A$: $a(trn') \in B(a(tst), r_a)$, where $r_a = \varrho_a(a(tst), a(trn))$. Then from definition of the closed ball we have $\varrho_a(a(tst), a(trn')) \leq \varrho_a(a(tst), a(trn))$.
- for all numerical attributes $a \in A$: $a(trn') \in [\min_a, \max_a]$, where $\min_a = \min(a(tst), a(trn))$, $\max_a = \max(a(tst), a(trn))$. Then, we have $|a(tst) - a(trn')| \leq |a(tst) - a(trn)|$. Thus, using

definition of metric for numerical attributes (normalised Euclidean metric) we have $\varrho_a(a(tst), a(trn')) = \frac{|a(tst) - a(trn')|}{a^{\max} - a^{\min}} \leq \frac{|a(tst) - a(trn)|}{a^{\max} - a^{\min}} = \varrho_a(a(tst), a(trn))$.

It means that for all attributes $a \in A$ we have $\varrho_a(a(tst), a(trn')) \leq \varrho_a(a(tst), a(trn))$. In consequence, we have the inequality between the global distances for *Agr* defined by sum of metrics $\varrho(tst, trn') = \sum_{a \in A} \varrho_a(a(tst), a(trn')) \leq \sum_{a \in A} \varrho_a(a(tst), a(trn)) = \varrho(tst, trn)$.

Adding multiplication factor (specified by a weight) for each attribute preserves the above inequality. In consequence, we have the same result also for aggregation function defined by weighted sum of metrics. $\square$

Hence, the examples that are distanced from the test example tst more than the training example trn cannot cause inconsistency of *g-rule* $(tst, trn, \{\varrho_a\}_{a \in A_{sym}})$. In consequence, we can substitute the set $trnSet$ in line 7 of Algorithm 4 by $N(tst, trnSet, k, \varrho)$.

The resulting classification algorithm RIONA is presented in Algorithm 5. The formal proof that this algorithm computes measure *LocStrength* is presented later in Theorem IV.2. This algorithm predicts the most common class among the training examples that are covered by the rules satisfied by a given test example and are in the specified neighbourhood. It is to be noted that in the argument of the algorithm, all pseudometrics are given (used for computation of the final pseudometric), but in the *g-rule* only pseudometrics for symbolic attributes are used (see Definition III.2 and note after it).

For every decision class, the RIONA algorithm computes the support set restricted to the neighbourhood $N(tst, k)$ rather than the whole support set of the maximally general rules covering a test object (as the RIA algorithm) in the following way. For every training object trn from the neighbourhood $N(tst, k)$ the algorithm constructs the rule *g-rule* $(tst, trn, \{\varrho_a\}_{a \in A_{sym}})$ based on the considered example trn and the test example tst . Then, it checks whether this *g-rule* is consistent with the remaining training examples from the neighbourhood $N(tst, k)$. If the local decision rule is consistent, then the training example trn used to construct the rule is added to the support set of the appropriate decision. Finally, the algorithm selects the decision with the support set of the highest cardinality.

IV. RELATIONSHIPS OF RIONA TO OTHER APPROACHES

Algorithm 5 (RIONA) is based on a combination of the kNN method with lazy rule induction. The only difference

⁵It should be observed that we use in the proof the assumption that pseudometrics used for grouping symbolic attributes are the same as the pseudometrics which compose the aggregated pseudometric used for measuring distance between examples. The analogous assumption is used for numerical attributes: real values are grouped using interval contained in the ball $B(a(tst), \varrho_a(a(tst), a(trn)))$ determined by the Euclidean metric. The same Euclidean metric (however normed) is used for components of the final pseudometric.

Algorithm 5: RIONA-classify($tst, trnSet, k, \{\varrho_a\}_{a \in A}$)

Input: test example tst , training set $trnSet$, positive integer k , family of pseudometrics for attributes $\{\varrho_a\}_{a \in A}$

Output: predicted decision for tst

```

1 begin
2    $\varrho = Agr(\{\varrho_a\}_{a \in A})$ 
3    $nSet = N(tst, trnSet, k, \varrho)$ 
4   foreach decision  $v \in V_d$  do
5      $supportSet(v) = \emptyset$ 
6   end
7   foreach  $trn \in nSet$  do
8      $v = d(trn)$ 
9     if  $isCons(g\text{-rule}(tst, trn, \{\varrho_a\}_{a \in A_{sym}}), nSet)$  then
10       $supportSet(v) = supportSet(v) \cup \{trn\}$ 
11    end
12  end
13  return  $\arg \max_{v \in V_d} |supportSet(v)|$ 
14 end

```

in comparison to Algorithm 1 (kNN) is in line 9 where we check the consistency of rule generated by training and testing example.

We have the following relationships between RIONA, RIA and kNN.

Proposition IV.1. For each test object tst

$$RIONA(tst, trnSet, k, \{\varrho_a\}_{a \in A}) = \begin{cases} RIA(tst, trnSet, \{\varrho_a\}_{a \in A_{sym}}) & \text{for } k \geq |trnSet| \\ 1NN(tst, trnSet, \varrho) & \text{for } k = 1, |N(tst, trnSet, k, \varrho)| = 1, \end{cases}$$

where 1NN is the nearest neighbour algorithm for $k = 1$ for pseudometric $\varrho = Agr(\{\varrho_a\}_{a \in A})$.

Proof. For $k \geq |trnSet|$, the neighbourSet in the RIONA algorithm (see Algorithm 5) is equal to $trnSet$. In this case, the RIONA algorithm works exactly as the RIA algorithm (see Algorithm 4).

For $k = 1$, $|N(tst, trnSet, 1, \varrho)| = 1$, the neighbourSet in the RIONA algorithm contains one training example⁶. In this case, consistency checking procedure could be eliminated. In consequence, the RIONA algorithm works exactly as 1NN (see Algorithm 1). $\square$

For the maximal neighbourhood, the RIONA algorithm works exactly as the RIA algorithm (and from Corollary III.5 as the algorithm based on the maximally general rules with *Strength* as a strategy for conflict resolution). On the other hand, taking a neighbourhood consisting of the single nearest

⁶Please note that the assumption about the cardinality of N is important. When $|N| > 1$, even for consistent training set, examples from the neighbourhood N (equally distanced from test example) will cause inconsistency whenever there are two objects in N with different decisions.

training example, we obtain the nearest neighbour algorithm. In this sense, RIONA is placed between the nearest neighbour classifier and the classifier based on maximally general rules. The choice of a small neighbourhood causes the algorithm to behave more like kNN classifier. The choice of a large neighbourhood causes the algorithm to behave more like a classifier based on inducing maximally general rules. Taking a larger but not the maximal neighbourhood can be seen as considering more specific rules instead of maximally general rules consistent with the training examples.

Now, we also show that we can look at the RIONA algorithm from other perspectives.

Theorem IV.2. For any test object tst , the classification result of the classifier from Eqn. 10 with $MaxRules = MaxRules(GenRules, trnSet)$, where $GenRules = GenRules(\{\{\varrho_a, a(tst)\}\}_{a \in A_{sym}})$, $\varrho = Agr(\{\varrho_a\}_{a \in A})$, we have $RIONA(tst, trnSet, k, \{\varrho_a\}_{a \in A}) = decLoc_{MaxRules}(tst, k, \varrho)$.

Proof. From Corollary III.5 $RIA(tst, trnSet, \{\varrho_a\}_{a \in A_{sym}}) = decision_{MaxRules}(tst)$. In fact, as we noticed it implied from more strong fact following from Theorem III.4 that the RIA algorithm computes measure *Strength* for $MaxRules = MaxRules(GenRules, trnSet)$, i.e. for each $v \in V_d$ $supportSet(v)$ (in line 11 of Algorithm 4) = *Strength*(v). The RIONA algorithm takes into account only training examples from $N(tst, trnSet, k, \varrho)$. Moreover, from Proposition III.6 examples consistent with $N(tst, trnSet, k, \varrho)$ are consistent with the whole training set, $trnSet$. At the same time, measure *LocStrength*(tst, v, k, ϱ) takes into account only training examples from the same neighbourhood $N(tst, trnSet, k, \varrho)$. In consequence, for each $v \in V_d$ $supportSet(v)$ (in line 13 of Algorithm 5) = *LocStrength*(tst, v, k, ϱ). This implies the equation of the theorem. $\square$

Theorem IV.3. For any test object tst , the classification result of the classifier from Eqn. 10 with $MaxRules = MaxRules(GenRules, N(tst, trnSet, k, \varrho))$, where $GenRules = GenRules(\{\{\varrho_a, a(tst)\}\}_{a \in A_{sym}})$, $\varrho = Agr(\{\varrho_a\}_{a \in A})$, we have $RIONA(tst, trnSet, k, \{\varrho_a\}_{a \in A}) = decLoc_{MaxRules}(tst, k, \varrho)$.

Proof. From Theorem IV.2 with $trnSet$ replaced by $N(tst, trnSet, k, \varrho)$ (it is in a sense treated as a new training set), we have $RIONA(tst, N(tst, trnSet, k, \varrho), k, f) = decLoc_{MaxRules}(tst, k, \varrho)$, where $f = \{\varrho_a\}_{a \in A}$. $MaxRules = MaxRules(GenRules, N(tst, trnSet, k, \varrho))$. Since $RIONA(tst, N(tst, trnSet, k, \varrho), k, f) = RIONA(tst, trnSet, k, f)$, this ends the proof. $\square$

To sum up, these two theorems give the following interesting corollary.

Corollary IV.4. For any test object tst , the results computed by the following classifiers are the same

- 1) $RIONA(tst, trnSet, k, \{\varrho_a\}_{a \in A})$,
- 2) $decLocMaxRules(tst, k, \varrho)$,
- 3) $decLocMaxLocalRules(tst, k, \varrho)$,
- 4) $decision_{MaxLocalRules}(tst)$ for new training set $trnSet' = N(tst, trnSet, k, \varrho)$.

where $\varrho = Agr(\{\varrho_a\}_{a \in A})$, $MaxRules = MaxRules(GenRules, trnSet)$, $MaxLocalRules = MaxRules(GenRules, N(tst, trnSet, k, \varrho))$, and $GenRules = GenRules(\{(\varrho_a, a(tst))\}_{a \in A_{sym}})$.

Proof. Equivalence of first three classifiers is implied directly by Theorems IV.2 and IV.3. It remains to prove equivalence of third and fourth classifiers. Note that $trnSet' = N(tst, trnSet, k, \varrho) = N(tst, N(tst, trnSet, k, \varrho), k, \varrho) = N(tst, trnSet', k, \varrho)$. Always holds $supportSet(r) \subseteq trnSet'$. So using previous equation $supportSet(r) \subseteq N(tst, trnSet', k, \varrho)$. Then $locSuppSet(r) = supportSet(r) \cap N(tst, trnSet', k, \varrho) = supportSet(r)$. We conclude that Eqn. 9 becomes Eqn. 7 and finally that Eqn. 10 becomes Eqn. 8. $\square$

In other words, we have the following conclusions. First, the RIONA algorithm computes the *LocStrength* measure (second algorithm above; see Eqn. 9). Second, the *LocStrength* measure is simply the *Strength* measure treating $N(tst, k)$ as the local training set (fourth algorithm). This fourth algorithm is the same algorithm as in Eqn. 8 with training set $trnSet$ replaced by local training sample $trnSet' = N(tst, trnSet, k, \varrho)$. Therefore the RIONA algorithm can be treated as an algorithm for computing all maximally general, consistent rules locally and using (locally) *Strength* for conflict resolution. In Table I a general comparison of these three algorithms is presented (we omit the third algorithm, which is very similar to the fourth). In Table II a comparison scheme for these three algorithms is presented (with explanation what is common and what is different in these algorithms).

RIONA	algorithm (2) based on the measure <i>LocStrength</i>	algorithm (4) based on the measure <i>Strength</i> counted locally
counting rules		
no need to count rules explicitly	counts <i>MaxRules</i> globally once at the beginning	counts <i>MaxRules</i> locally for each test case
counting support		
counts support using lazy local rules	counts support locally	counts support locally

Table I

A GENERAL COMPARISON OF THREE ALGORITHMS FROM COROLLARY IV.4: ALGORITHM (1) RIONA, ALGORITHM (2) BASED ON THE MEASURE *LocStrength* AND ALGORITHM (4) BASED ON THE MEASURE *Strength* COUNTED LOCALLY.

A. RIONA and rules

The RIONA algorithm has properties of instance-based classifiers and rule-based classifiers. There are some aspects of rule-based classifiers which are more preferred for users than instance-based classifiers even at the expense of decreasing the quality of classification. One of these important aspects is the

RIONA	algorithm (2) based on the measure <i>LocStrength</i>	algorithm (4) based on the measure <i>Strength</i> counted locally
Global input: $trnSet, k \in \mathbb{N}$		
1.	count <i>MaxRules</i> for $trnSet$	
Input: test case tst		
2. $nSet = N(tst, k)$		
3.	$RuleBase = MaxRules$	count (locally) $MaxRules(nSet)$ $RuleBase = MaxRules(nSet)$
4.	consider rules from $RuleBase$ with premise satisfied by tst	
5. for each decision d		
6. consider $trn \in nSet$ with decision d	consider rules from step 4 with decision d	
7. count the number of trn from step 6 forming consistent rules with tst		count the number of $trn \in nSet$ supporting rules from step 6
8. choose the decision with the maximal count (maximally supported)		

Table II

A COMPARISON SCHEME OF THREE ALGORITHMS FROM COROLLARY IV.4: ALGORITHM (1) RIONA, ALGORITHM (2) BASED ON THE MEASURE *LocStrength* AND ALGORITHM (4) BASED ON THE MEASURE *Strength* COUNTED LOCALLY.

possibility to interpret rules by a human, non-computer science expert. He or she can verify whether the discovered knowledge in such rules is non-trivial, true in fact and revealing new aspects of the regarded problem. A rule contains an explanation for taking the particular decision easily understandable by a human.

We assume here that the parameter k in the RIONA algorithm is fixed (possibly learned as described in [13]). Now, let us focus on algorithm (4) from the previous subsection. At first sight, the direct computation of *MaxLocalRules* may seem very expensive and infeasible because for each test case tst it is necessary to compute the local complete set of consistent and maximally general decision rules. However, let us note that the size of the local training sample compared to the size of the whole training sample is significantly reduced from $n = |trnSet|$ to k (if we assume that the size of N is k ; see previous notes related to this issue). Thus the total cost of computation of (global or local) *MaxRules* is reduced from $O(2^n)$ to $O(m \cdot 2^k)$, where m is the number of test objects. We present this approach not only from a theoretical point of view. Such an algorithm could be used when an explanation of the decision undertaken by the classifier is required. In this sense, the RIONA algorithm has features of quick lazy learning algorithm and rule algorithm, i.e. its parameters can be translated into rules.

Moreover, it seems possible to extend algorithm (4) to build all rules globally once at the beginning analogously to algorithm (2) from Corollary IV.4 with such difference that the rules would base on the local neighbourhood. Such rules would imitate the behaviour of the RIONA algorithm. There are some advantages of such an approach. Firstly, the explanation for the specific test object in the form of a set of rules could be given quickly. Secondly, all possible rules generated at the beginning could be verified whether the discovered knowledge is really useful.

We give here only an informal description of how such rules could be generated. The idea is similar to algorithm

(4) from Corollary IV.4. We could simply treat each training example as a test example and generate *MaxRules* locally for each training case. It could be seen as specific local reducts calculation (i.e. reducts calculated in the process of generation of maximally general rules for a given object; see e.g. [26], [32], [6], [5]). Normally, in constructing local reducts, discernibility should be preserved for objects with different decisions. Here, we would require that only objects with a different decision and distanced not more than k should be discerned.

V. CONCLUSION

Theoretical results explain the RIONA classifier's relationships both instance- and rule-based classifiers. In particular, we showed the theoretical equivalence of the RIONA algorithm with the algorithm generating all consistent and maximally general rules in a training set consisting of the neighbourhood of the test case. Consequently, the RIONA classifier can be represented by a relatively small rule set, with rules easily understandable by a human. It could be used in the situation when an explanation or justification of the derived decision is important.

To sum up, the RIONA algorithm combining instance- and rule-based approaches, which is both efficient and effective (in classification) returns classifiers, which have the property of explainability.

REFERENCES

- [1] Rseslib 3: Rough set and machine learning open source in Java. <http://rseslib.mimuw.edu.pl>
- [2] Amina Adadi and Mohammed Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 1:2138–52160, 2018.
- [3] David W. Aha, editor. *Lazy Learning*. Springer, Dordrecht, 1st edition, 1997.
- [4] Ammar Almasri, Erbug Celebi, and Rami S. Alkhaldeh. EMT: Ensemble Meta-Based Tree Model for Predicting Student Performance. *Scientific Programming*, 2019:Article No. 3610248, 1–13, 2019.
- [5] Jan G. Bazan. Discovery of Decision Rules by Matching New Objects Against Data Tables. In *Rough Sets and Current Trends in Computing (RSCTC 1998)*, pages 521–528. Heidelberg, 1998. Springer.
- [6] Jan G. Bazan, Hung Son Nguyen, Sinh Hoa Nguyen, Piotr Synak, and Jakub Wróblewski. Rough Set Algorithms in Classification Problem. In Lech Polkowski et al. [27], pages 49–88.
- [7] Jan G. Bazan and Marcin Szczuka. RSES and RSESlib – A Collection of Tools for Rough Set Computations. In *Rough Sets and Current Trends in Computing (RSCTC 2001)*, pages 106–113. Heidelberg, 2001. Springer.
- [8] Rodrigo de Oliveira Almeida and Guilherme Targino Valente. Predicting metabolic pathways of plant enzymes without using sequence similarity: Models from machine learning. *The Plant Genome*, 13(3):e20043, 2020.
- [9] Nilanjan Dey, Samarjeet Borah, Rosalina Babo, and Amira S. Ashour, editors. *Social Network Analytics: Computational Research Methods and Techniques*. Academic Press, London, 1st edition, 2019.
- [10] Filip Karlo Došliović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0210–0215. Croatian Society MIPRO, 2018.
- [11] Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrac. *Foundations of Rule Learning*. Cognitive Technologies. Springer, Heidelberg, 2012.
- [12] Grzegorz Góra. *Combining instance-based learning and rule-based methods for imbalanced data*. PhD thesis, University of Warsaw, Warsaw, 2022. https://www.mimuw.edu.pl/sites/default/files/gora_grzegorz_rozprawa_doktorska.pdf
- [13] Grzegorz Góra and Arkadiusz Wojna. RIONA: A Classifier Combining Rule Induction and K-nn Method with Automated Selection of Optimal Neighbourhood. In *Proceedings of the 13th European Conference on Machine Learning (ECML 2002)*, pages 111–123. Heidelberg, 2002. Springer-Verlag.
- [14] Grzegorz Góra and Arkadiusz Wojna. RIONA: A New Classification System Combining Rule Induction and Instance-Based Learning. *Fundamenta Informaticae*, 51(4):369–390, 2002.
- [15] Lacrimioara Grama and Corneliu Rusu. Choosing an accurate number of mel frequency cepstral coefficients for audio classification purpose. In *Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis (ISPA 2017)*, pages 225–230, 2017.
- [16] Lacrimioara Grama and Corneliu Rusu. Adding audio capabilities to TIAGo service robot. In *2018 International Symposium on Electronics and Telecommunications (ISETC)*, pages 1–4, 2018.
- [17] Jerzy W. Grzymala-Busse. LERS-A System for Learning from Examples Based on Rough Sets. In Roman Słowiński, editor, *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, pages 3–18. Springer, Dordrecht, 1992.
- [18] Hani Hagaras. Toward Human-Understandable, Explainable AI. *Computer*, 51(9):28–36, 2018.
- [19] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, NY, 2nd edition, 2009.
- [20] Ryszard S. Michalski, Igor Mozetic, Jiarong Hong, and Nada Lavrac. The Multi-Purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains. In *Proceedings of the 5th AAAI National Conference on Artificial Intelligence*, pages 1041–1045. AAAI Press, 1986.
- [21] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997.
- [22] Hung Son Nguyen. Approximate Boolean Reasoning: Foundations and Applications in Data Mining. In James F. Peters and Andrzej Skowron, editors, *Transactions on Rough Sets V*, pages 334–506. Springer, Heidelberg, 2006.
- [23] Hung Son Nguyen and Sinh Hoa Nguyen. Discretization Methods in Data Mining. In Lech Polkowski and Andrzej Skowron, editors, *Rough Sets in Knowledge Discovery I: Methodology and Applications*, volume 18 of *Studies in Fuzziness and Soft Computing*, pages 451–482. Heidelberg, 1998. Physica-Verlag.
- [24] Sinh Hoa Nguyen. Regularity Analysis and its Applications in Data Mining. In Polkowski et al. [27], pages 289–378.
- [25] Son H. Nguyen and Andrzej Skowron. Quantization Of Real Value Attributes – Rough Set and Boolean Reasoning Approach. In *Proceedings of the 2nd Joint Annual Conference on Information Sciences (JCIS 1995)*, pages 34–37, 1995.
- [26] Zdzisław Pawlak and Andrzej Skowron. A Rough Set Approach to Decision Rules Generation. In *Proceedings of the Workshop W12: The Management of Uncertainty at the 13th International Joint Conference on Artificial Intelligence (IJCAI 1993)*, pages 114–119. Chambéry, 1993. Morgan Kaufmann.
- [27] Lech Polkowski, Shusaku Tsumoto, and Tsau Y. Lin, editors. *Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems*, volume 56 of *Studies in Fuzziness and Soft Computing*. Physica-Verlag, Heidelberg, 2000.
- [28] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, Hoboken, NJ, 4th edition, 2021.
- [29] Corneliu Rusu and Lacrimioara Grama. Recent developments in acoustical signal classification for monitoring. In *2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE)*, pages 110, 2017.
- [30] Claude Sammut and Geoffrey I Webb, editors. *Encyclopedia of Machine Learning and Data Mining*. Springer, USA, 2nd edition, 2017.
- [31] Andrzej Skowron and Cecylia Rauszer. The Discernibility Matrices and Functions in Information Systems. In Roman Słowiński, editor, *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, pages 331–362. Springer, Dordrecht, 1992.
- [32] Jakub Wróblewski. Covering with Reducts – A Fast Algorithm for Rule Generation. In *Rough Sets and Current Trends in Computing (RSCTC 1998)*, pages 402–407. Heidelberg, 1998. Springer.

87%

SIMILARITY INDEX

PRIMARY SOURCES

1	www.depotuw.ceon.pl Internet	9478 words — 83%
2	dokumen.pub Internet	49 words — < 1%
3	citeseerx.ist.psu.edu Internet	47 words — < 1%
4	nozdr.ru Internet	47 words — < 1%
5	thesis.lib.ncu.edu.tw Internet	36 words — < 1%
6	zh.scribd.com Internet	35 words — < 1%
7	epdf.pub Internet	32 words — < 1%
8	www.ijcai.org Internet	31 words — < 1%
9	repo.pw.edu.pl Internet	29 words — < 1%
10	perso.ens-lyon.fr Internet	

28 words — < 1%

11 repositories.lib.utexas.edu
Internet

25 words — < 1%

12 onlinelibrary.wiley.com
Internet

24 words — < 1%

13 Małgorzata Przybyła-Kasperek. " Study of selected methods for balancing independent data sets in -nearest neighbors classifiers with Pawlak conflict analysis ", Applied Soft Computing, 2022
Crossref

16 words — < 1%

14 bcpw.bg.pw.edu.pl
Internet

15 words — < 1%

15 ftp.math.utah.edu
Internet

13 words — < 1%

16 Andrzej Skowron, Andrzej Jankowski. "Interactive computations: toward risk management in interactive intelligent systems", Natural Computing, 2015
Crossref

10 words — < 1%

17 ipfs.io
Internet

8 words — < 1%

18 www.mimuw.edu.pl
Internet

8 words — < 1%

19 "Rough Sets and Current Trends in Computing", Springer Science and Business Media LLC, 2004
Crossref

6 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	OFF	EXCLUDE MATCHES	OFF