Fundamenta Informaticae XX (2002) 1–22
IOS Press

RIONA: A New Classification System Combining Rule Induction and Instance-Based Learning

Grzegorz Góra*
Institute of Informatics, Warsaw University
ul. Banacha 2, 02-097 Warszawa, Poland
ggora@mimuw.edu.pl

Arkadiusz Wojna*
Institute of Informatics, Warsaw University
ul. Banacha 2, 02-097 Warszawa, Poland
wojna@mimuw.edu.pl

Abstract. The article describes a method combining two widely-used empirical approaches to learning from examples: rule induction and instance-based learning. In our algorithm (RIONA) decision is predicted not on the basis of the whole support set of all rules matching a test case, but the support set restricted to a neighbourhood of a test case. The size of the optimal neighbourhood is automatically induced during the learning phase. The empirical study shows the interesting fact that it is enough to consider a small neighbourhood to achieve classification accuracy comparable to an algorithm considering the whole learning set. The combination of k-NN and a rule-based algorithm results in a significant acceleration of the algorithm using all minimal rules. Moreover, the presented classifier has high accuracy for both kinds of domains: more suitable for k-NN classifiers and more suitable for rule based classifiers.

Keywords: machine learning, instance-based learning, rule induction, nearest neighbour method

1. Introduction

Inductive concept learning is a process of synthesis of concept description from examples. Many techniques of inductive concept learning have been developed so far [25]. They include induction of decision

*Address for correspondence: Institute of Informatics, Warsaw University, ul. Banacha 2, 02-097 Warszawa, Poland

trees (see e.g. [27]), rule induction (see e.g. [23]), instance-based learning (see e.g. [1]), neural networks (see e.g. [5]) and others.

Empirical comparison of these approaches shows that each performs well for some, but not all, domains. A great progress has been made in multistrategy learning to combine these approaches in order to construct a classifier that has properties of two or more techniques. Although the problem of inductive generalisation has no general solution (what is known as the conservation law for generalisation performance [28]), the goal is to increase the average accuracy for the real-world domains at the expense of accuracy decreasing for the domains that never occur in practice.

There are domains for which instance-based learning [1], [8], [9], [11] and rule induction [7], [24] achieve significantly different accuracy. Generally, instance-based approach is more accurate for numerical domains, while rule induction is better for domains with symbolic attributes or with attributes irrelevant with respect to the decision. In the instance-based learning a common approach is the k-nearest neighbours (*k-NN*) method.

In this paper we present a multi-strategy learning approach combining the rule induction and the nearest neighbour technique. There has been a lot of work done in this area (see e.g. [10], [13], [19], [20]). Our algorithm considers all minimal decision rules, i.e. the most general rules consistent with training examples. It simulates classification based on the most frequent decision in the support set of minimal rules covering a test object. The main idea is that the support set is limited to a neighbourhood of a test example. The neighbourhood consists of either objects within some distance from a test example or a number of objects closest to a test example (like in *k-NN* method). The appropriate size of a neighbourhood to be taken for classification is automatically induced during the process of learning. The crucial empirical observation is that taking a neighbourhood much smaller than the whole training set preserves or even improves accuracy. It enables both to induce optimal neighbourhood during learning phase and to classify objects effectively.

The notion of the rule defined in the paper realises a new approach to dealing with attribute value domain. The rules are generated during the classification process and each generated rule relates to a particular tested object. A rule descriptor is formulated either as an interval inclusion for a real-value attribute or as an inclusion in a value subset for a symbolic attribute. Each descriptor is generated in the context of a particular attribute value of a test object and corresponds to local grouping of values.

The paper is organised as follows. Section 2 describes a number of works related to the subject. Section 3 outlines the main features of two techniques that are the most relevant to this work, i.e. rule induction and instance based learning exemplified by k-nearest neighbours method. Our algorithm, combining these approaches, is presented in Section 4. Section 5 provides experimental results evaluating the accuracy and speed of the presented system. Section 6 concludes the paper with a brief summary and discussion of possible directions for future research.

2. Related Work

In recent literature there has been a number of works combining instance-based and decision rule induction methods.

RISE system [10] is based on unification of these two methods. The difference between *RISE* system and our approach is that *RISE* selects the decision for a test object on the basis of the closest rule. First, *RISE* generates decision rules. At the beginning instances are treated as maximally specific rules and

these rules are then gradually generalised as long as global leave-one-out accuracy is improving. An object is classified according to the closest rule. The distance between an object and a rule is measured with the metric combining normalised Manhattan metric[1] for numerical attributes and the Simple Value Difference Metric (SVDM) for symbolic attributes.

An approach more similar to our method is presented in *DeEPs* [19] and *DeEPsNN* [20]. The first difference is that *DeEPs* uses different form of rule conditions and different criteria for rule selection. *DeEPs* classifies objects on the basis of all rules that have high frequency-changing rate (a measure similar to confidence). While classifying a test object the system computes the support set using all rules with high frequency-changing rate and selects the most frequent decision in the support set. In our system a computed support set is limited to a certain neighbourhood of a test object. *DeEPsNN* combines *3-NN* and *DeEPs*: if a certain fixed neighbourhood of a test object covers at least one training object, *3-NN* is applied, otherwise *DeEPs* is used.

In [2] an algorithm with lazy rule induction approach is presented. It computes the whole support set of the minimal rules covering a test object in the following way. For each training object the algorithm constructs the local rule containing the conditions common for the test and the training object and checks whether the training objects supporting the constructed local rule are in the same decision class. Then the algorithm selects the decision most frequent in the support set. This algorithm treats all attributes as symbolic. We generalised this algorithm for symbolic attributes and extended it to numerical attributes. Our approach relates to discretisation of numerical attributes and the methods for grouping attribute values [15]. It does not require any prior discretisation. A similar approach for numerical attributes was presented in [18]. However, in our approach discretisation is done implicitly during classification locally for a test example. Also our approach is parameterised by the choice of a metric for non-ordered attributes.

A detailed study of *k-NN* algorithms is presented in [31]. In particular, that paper describes research on selection of the optimal value of k. The experiments presented in that paper showed that the accuracy of *k-NN* is insensitive to the exact choice of k when the optimal k is large enough. Different methods for adapting the value of k locally within different parts of the input space have also been investigated. The local selection of k improves accuracy for data that contain noise or irrelevant features.

Our approach combines the idea used in [2] (extended as described above) with *k-NN* method in such a way that it considers the local rules only for the training examples from the k-nearest neighbourhood of a test example. The distance is measured with the metric used in *RISE* [10]. Moreover, the algorithm searches for the global optimal value k during the learning phase. This combination improves the accuracy of *k-NN* classifiers with a fixed value k and helps to reach accuracy comparable to rule-based classifiers in cases when accuracy of *k-NN* method is low.

3. Preliminaries and Definitions

We assume that a training set, denoted in the paper $trnSet$, is a finite set of examples. Each example is described by a finite set of attributes (features) $A \cup \{d\}$, i.e. $a : trnSet \to V_a$ for $a \in A \cup \{d\}$, where $d \notin A$ denotes the decision attribute and V_a is a value domain of the attribute a. The domain of symbolic (discrete-valued) attribute is a finite set, while the domain of a numerical (real-valued) attribute is an

[1] Manhattan metric is also called city metric defined for $\mathbf{x} = (x_1, ..., x_k)$, $\mathbf{y} = (y_1, ..., y_k)$ as $\sum_i |x_i - y_i|$.

interval. We denote by $Class(v)$ the subset of all training examples with a decision v. We also assume that $V_d = \{1, ..., |V_d|\}$.

3.1. Minimal and Lazy Rule Induction

Rule induction algorithms induce decision rules from a training set. Those rules can be presented in the form $IF\ (t_1 \wedge t_2 \wedge ... \wedge t_k)\ THEN\ (d = v)$, where t_i is a condition concerning an attribute a_i, $i = 1, 2..., k$; d is the decision, and v is its value. The commonly used conditions for symbolic attributes are equations $attribute = value$, while for numerical attributes are specified by interval inclusions, e.g.:

$$IF\ (a_1 = 2 \wedge a_3 \in [3; 7] \wedge a_6 = 5)\ THEN\ (d = 1)$$

If a problem domain contains some numerical attributes then the appropriate intervals can be obtained by applying discretisation (see e.g. [26]). The consequent $(d = v)$ denotes a decision value that is assigned to an object if it satisfies the premise of the rule.

From the knowledge discovery perspective, an important problem is to compute the complete set of consistent and minimal decision rules (see e.g. [29]), i.e. all rules (matched at least by one training example) that are maximally general and consistent with a training set. In order to obtain an efficient and accurate classifier many systems compute rule sets satisfying these conditions in an approximate way (see e.g. [3]). However we consider the rule set defined above and denote it by $MinRules$. In order to discover $MinRules$, rough set methods [29] can be used.

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. [24]). A common approach is to use a measure for conflict resolving and decision with the highest measure value is chosen. In this paper we focus on commonly used measures that are presented below.

$$Strength(tst, v) = \left| \bigcup_{r \in MatchRules(tst,v)} supportSet(r) \right| \tag{1}$$

$$NormStrength(tst, v) = \frac{Strength(tst, v)}{|Class(v)|} \tag{2}$$

where v denotes the v-th decision ($v = 1, ..., |V_d|$), tst is a test example, $supportSet(r)$ is a set of training examples matching the rule r, $MatchRules(tst, v)$ is a subset of minimal rules $MinRules$, whose premise is satisfied by tst and the consequent is a decision v. The measure $Strength$ counts the number of training examples that were covered by the minimal rules with the decision matching a test example tst. The second measure is just $Strength$ measure normalised by the decision class size.

Conflict resolving is an intensively studied topic in data mining, the reader is referred to [24] for more details and other measures. Since now, we use in the paper $NormStrength$ as a measure for conflict resolving, but in the experiments both measures were compared (see Section 5).

The minimal rule induction classifier based on the measure $Strength$ predicts the decision that is most frequent in the set of training examples covered by rules matched by a test example, i.e.:

$$decision_{MinRules}(tst) = \arg\max_{v \in V_d} NormStrength(tst, v)$$

Algorithms for computing all minimal rules ($MinRules$) are very time-consuming, especially when the number of training objects or attributes is large. This is due to the fact that the size of the $MinRules$ set can be exponential with respect to the size of the training set. In practice, as it was mentioned above, approximation algorithms are often applied to obtain the rule set that is not necessarily complete.

Another approach can be based on construction of algorithms that do not require calculation of the decision rule set before classification of new objects. These are memory based (lazy concept induction) algorithms. An example of such an algorithm is presented in [2]. 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 Equation (1) and (2) for every test object without computing all minimal rules ($MinRules$). Below we describe a version of this algorithm generalised for symbolic attributes and extended to the case of numerical attributes.

In the original form in [2] the algorithm worked only with symbolic attributes and used the notion of a rule with the definition of an attribute descriptor based on the Hamming distance:

Definition 3.1. For objects tst, trn we denote by $rule^{H}_{tst}(trn)$ the local rule with the decision $d(trn)$ and the following conditions t for each symbolic attribute a:

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

where * denotes any value (such a condition is always true).

The extended version of the algorithm uses more specific conditions to form a local rule instead of the "star" condition in case when attribute values of the examples differ. In the definition above the star represents the group of all values from the attribute domain. However, we noticed that a proper subset of all attribute values can be more relevant for the classification. Then, we propose the following generalisation of Definition 3.1 for both numerical and symbolic attributes:

Definition 3.2. For objects tst, trn we denote by $rule_{tst}(trn)$ the local rule with the decision $d(trn)$ and the following conditions t for each attribute a:

$$t = \begin{cases} a \in [min(a(tst), a(trn)), max(a(tst), a(trn))] & \text{when } a \text{ is numerical} \\ a \in B\,(a(tst), \delta_a(a(tst), a(trn))) & \text{when } a \text{ is symbolic} \end{cases}$$

where $B(c, R)$ is a ball centered in c with radius R and δ_a is a measure of attribute value similarity.

For linearly ordered attributes conditions are represented in the form of interval inclusion, i.e. they require from a value to lay between the attribute values of the examples tst, trn forming the rule. Non-ordered attributes are treated differently. For such an attribute a metric that defines distances among the values of an attribute is required to be defined. The condition selects the group of values that are distanced from the attribute value of the example tst no more than the attribute value of the example trn, i.e. they can be represented by a ball centered in the attribute value of tst with the radius equal to the distance between the attribute values of tst and trn. If δ_a in Definition 3.2 is the Kronecker delta[2] ($\delta_a(x, y) = 1$ if $x = y$ and 0 otherwise) then the conditions for symbolic attributes are equivalent to the

[2]It relates to the Hamming distance between attribute vector values.

condition used in Definition 3.1:, i.e. when $a(tst) = a(trn)$ then the condition is $a = a(trn)$, otherwise the condition is the "star" condition.

The conditions are chosen in such a way, that both the training and the test example satisfy the rule and the conditions are maximally specific, it means that making the interval smaller for a numerical attribute or making the radius smaller for a symbolic attribute will cause the example trn not to satisfy the rule.

As an example let us consider the following training set:

Object	Age	Weight	Gender	BloodGroup (BG)	Diagnosis
trn_1	35	90	M	A	Sick
trn_2	40	65	F	AB	Sick
trn_3	45	68	F	AB	Healthy
trn_4	40	70	M	AB	Healthy
trn_5	45	75	M	B	Sick
trn_6	35	70	F	B	Healthy
trn_7	45	70	M	0	Healthy
tst	50	72	F	A	?

Age and Weight are numerical attributes while Gender and BG are symbolic non-ordered attributes. Considering the SVDM metric for the attribute BG we obtain the following distances among the values of the attribute:

$$\delta_{BG}(A, AB) = \left|1 - \frac{1}{3}\right| + \left|0 - \frac{2}{3}\right| = \frac{4}{3}, \delta_{BG}(A, B) = 1, \delta_{BG}(A, 0) = 2$$

Then for the training object trn_1 the $rule_{tst}(trn_1)$

$$\text{if } (Age \in [35; 50] \wedge Weight \in [72; 90] \wedge BG = A) \text{ then } Diagnosis = Sick$$

is consistent because no other object from the training set satisfies the premise of this rule. But for the training object trn_2 the $rule_{tst}(trn_2)$

$$\text{if } (Age \in [40; 50] \wedge Weight \in [65; 72] \wedge Gender = F \wedge BG \in \{A, AB, B\}) \text{ then } Diagnosis = Sick$$

is inconsistent because the object trn_3 satisfies the premise of the rule and has a different decision.

We have the following relation between $MinRules$ and local rules.

Proposition 3.1. The premise of the $rule_{tst}(trn)$ implies the premise of a rule from the set $MinRules$ if and only if the $rule_{tst}(trn)$ is consistent with the training set.

This is a version of a proposition from [2] for the generalised local rules presented in this paper and follows from the construction of local rules and $MinRules$. This proposition shows that instead of computing the support sets for rules from $MinRules$ covering a new test case, it is sufficient to generate the local rules for all training examples and then check their consistency with the training set. It is done by the lazy rule induction algorithm (*RIA*) presented below.

Algorithm 3.1. $isConsistent(\alpha \Rightarrow (d = v), verifySet)$

for each $trn \in verifySet$
  if $d(trn) \neq v$ **and** trn satisfies α **then return** $false$
return $true$

Algorithm 3.2. $RIA(tst)$

1.**for** each decision $v \in V_d$
2. $supportSet(v) = \emptyset$
3. **for** each $trn \in trnSet$ with $d(trn) = v$
4. **if** $isConsistent(rule_{tst}(trn), trnSet)$ **then**
5. $supportSet(v) = supportSet(v) \cup \{trn\}$
6.$RIA = \arg\max_{v \in V_d} |supportSet(v)|$

The function $isConsistent(r, verifySet)$ checks if a local rule r is consistent with a $verifySet$. For every decision class the algorithm 3.2 computes the whole support set of the minimal rules covering a test object tst in the following way. For every training object trn it constructs the local $rule_{tst}(trn)$ based on the examples tst and trn. Then it checks whether the local $rule_{tst}(trn)$ is consistent with the remaining training examples, i.e. if all the training examples satisfing the $rule_{tst}(trn)$ are labeled with the same decision as the considered training example trn. If the local $rule_{tst}(trn)$ is consistent then the training example trn is added to the support set of the appropriate decision. Finally, the algorithm selects the decision with the support set of the highest cardinality.

From Proposition 3.1 it can be concluded that algorithm RIA computes measure $Strength$ and thus the results of the mentioned algorithm are equivalent to the results of the algorithm based on calculating $MinRules$ and using the measure $Strength$ as a strategy for conflict resolving (see [2]).

Corollary 3.1. For any test object tst, $RIA(tst) = decision_{MinRules}(tst)$.

If one compares *RIA* to the minimal rule induction algorithm it considers only the decision rules that can be involved in the classification of a given test object.

The time complexity of the algorithm RIA for a single test object is $O(n^2)$, where n is the number of objects in training data. This gives us a classification algorithm working in time $O(mn^2)$, where m is the number of test cases, which is far more efficient than generating $MinRules$. However, for problem domains with quite a large number of examples (like *letter* or *satimage*) this time complexity is still too high to be used in practice. One of the motivations behind our work is to reduce this complexity. As a result we apply some modifications to this algorithm. This will be described in Section 4.

For more details related to the original version of this algorithm the reader is referred to [2].

3.2. Instance-Based Learning

Reasoning and learning from cases is based on the concept of similarity. Given a number of examples the decision class for a new test case is inferred from the nearest stored example (or k nearest examples)

in a sense of a similarity measure. The performance of instance based learning depends critically on the similarity measure used, which is often estimated by a distance (a metric). For numerical domains (i.e., domains where all the features are real-valued), Manhattan and Euclidean distances are natural candidates. For symbolic attributes many instance based learning systems use Hamming distance, measuring number of attributes for which objects differ. For domains with both numerical and symbolic attributes a combination of these approaches may be used.

Below we present a metric that combines normalised Manhattan metric for numerical attributes with SVDM metric for symbolic attributes [10]. If there are m attributes, the distance between two instances $x = (a_1(x), a_2(x), ..., a_{|A|}(x), d(x))$ and $y = (a_1(y), a_2(y), ..., a_{|A|}(y), d(y))$ can be defined by:

$$\varrho(x, y) = \sum_{a \in A} \delta_a(x, y)$$

where $\delta_a(\cdot, \cdot)$ is a measure of attribute value similarity. For numerical attributes a commonly used approach for attribute value similarity is to normalise the value difference by its largest observed value difference:

$$\delta_a(x, y) = \left| \frac{a(x) - a(y)}{a^{\max} - a^{\min}} \right| \tag{3}$$

where $a^{\max}$ and $a^{\min}$ are the maximal and the minimal value for an attribute a among training examples. For symbolic attributes a more informative alternative than Hamming distance is SVDM metric:

$$\delta_a(x, y) = \sum_{v \in V_d} |P(Class(v)|a(x)) - P(Class(v)|a(y))| \tag{4}$$

SVDM (see e.g. [10]) considers two symbolic values to be similar if they have similar decision distribution, i.e. if they correlate similarly with the decision. Different variants of this metric have been successfully used previously (see e.g. [30], [8], [4]). In the presented paper we consider the metric that combines the presented similarity measures: (3) for numerical attributes and (4) for symbolic attributes.

The commonly used instance-based algorithm is the k nearest neighbours classification algorithm (*k-NN*). It is based on the assumption that examples that are closer in the instance space have the same decision. Hence, test examples are classified with the decision class like the decision most common in the set of k nearest neighbours from the training set.

4. Rule Induction with Optimal Neighbourhood Algorithm (RIONA)

Instead of considering all training examples in building a support set, like in the algorithm *RIA*, we can limit it to a certain neighbourhood of a test example. The intuition behind it is that training examples far from a test object are less relevant for classification than closer examples.

We consider two classes of neighbourhoods defined below.

Definition 4.1. For each test example tst we define $S(tst, k)$ as the set of k training examples that are most similar to tst according to similarity measure ϱ[3].

[3]In case when more than one example has an equal distance from the object tst to the k-th nearest example, all of them are added to $S(tst, k)$ (then the set $S(tst, k)$ contains more than k examples).

Definition 4.2. For each test example tst we define $B(tst, R) = \{trn \in trnSet : \varrho(tst, trn) \leq R\}$.

The former neighbourhood is similar to the one used in the *k-NN* algorithm. The latter neighbourhood is a set of objects distanced from tst no more than R according to a distance measure ϱ. From now on, we use in the paper $S(tst, k)$ neighbourhood, but $B(tst, R)$ neighbourhood can be used as well. We studied both classes of a neighbourhood in parallel and the empirical difference between them will be discussed in Section 5 presenting experimental results.

Now we are ready to present an approach to induction that is a kind of combination of case-based learning (see Section 3.2) and lazy minimal rule induction (see Section 3.1). The main idea is that we apply strategy for conflict resolving with measures (1) and (2) slightly changed in the following way:

$$NStrength(tst, v) = \left| \bigcup_{r \in MatchRules(v)} supportSet(r) \cap S(tst, k) \right| \tag{5}$$

$$NormNStrength(tst, v) = \frac{NStrength(tst, v)}{|Class(v)|} \tag{6}$$

where notation is the same as in equations (1), (2). The difference is that we consider only those examples covered by the rules matching a test object that are in a specified neighbourhood of the test example. As it was mentioned above we considered also the neighbourhood $B(tst, R)$. The appropriate measures related to this neighbourhood could be obtained by substituting $S(tst, k)$ with $B(tst, R)$ in equation 5.

In the classification process we assume that parameter of the neighbourhood is fixed, i.e. a number k for $S(tst, k)$ neighbourhood and a value R for $B(tst, R)$ neighbourhood. The proper size of the neighbourhood (the parameter k or R) is found in the learning phase (see Section 4.1).

Given a set $MinRules$ the above measures can be calculated by limiting the support sets of the rules matching a test example to the specified neighbourhood of a test example. Thus the minimal rule induction algorithm with the modified measure can be used here. Again, we used the lazy minimal rule induction methodology.

To implement this approach we used a modified version of Algorithm 3.2. First, in the line 3 of the algorithm only examples $trn \in S(tst, k)$ should be considered. Furthermore, it is not necessary to consider all the examples from the training set to check the consistency of the $rule_{tst}(trn)$. Please note that from Definition 3.2 we have that:

Proposition 4.1. If trn' satisfies $rule_{tst}(trn)$ then $\varrho(tst, trn') \leq \varrho(tst, trn)$.

Hence, the examples that are distanced from the test example tst more than the training example trn can not cause inconsistency of $rule_{tst}(trn)$.

The resulting classification algorithm is presented below. It predicts the most common class among the training examples that are covered by the rules satisfied by a test example and are in the specified neighbourhood.

Algorithm 4.1. $RIONA(tst)$

$neighbourSet = S(tst, k)$
for each decision $v \in V_d$
  $supportSet(v) = \emptyset$
  for each $trn \in neighbourSet$ with $d(trn) = v$
    if $isConsistent(rule_{tst}(trn), neighbourSet)$
      then $supportSet(v) = supportSet(v) \cup \{trn\}$
$RIONA(tst) = \arg\max_{v \in V_d} |supportSet(v)|$

For every decision class the algorithm *RIONA* computes not the whole support set of the minimal rules covering a test object (as the algorithm *RIA*), but restricted to the neighbourhood $S(tst, k)$ in the following way. For every training object trn that belongs to the neighbourhood $S(tst, k)$ the algorithm constructs the local $rule_{tst}(trn)$ based on the considered example trn and the test example tst. Then, it checks whether the local $rule_{tst}(trn)$ is consistent with the remaining training examples from the neighbourhood $S(tst, k)$. If the local rule is consistent then the training example trn that was 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.

We have the following dependencies between *RIONA*, *RIA* and the nearest neighbour algorithm.

Proposition 4.2. For each test object tst

$$RIONA(tst) = \begin{cases} RIA(tst) = decision_{MinRules}(tst) & \text{for } k = \infty \\ \textit{1-NN}(tst) & \text{for } k = 1 \end{cases}$$

where *1-NN* is the nearest neighbour algorithm.

For the maximal neighbourhood the algorithm *RIONA* works exactly as *RIA* algorithm (and minimal rule induction algorithm with $Strength$ as a strategy for conflict resolving). On the other hand taking a neighbourhood as the single nearest training example we obtain the nearest neighbour algorithm. In this sense *RIONA* belongs between the nearest neighbour and the minimal rule induction classifier. The choice of a small neighbourhood causes the algorithm to behave more like *k-NN* classifier and the choice of a large neighbourhood causes the algorithm to behave more like a common minimal rule induction classifier. 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.

4.1. Selection of Optimal Neighbourhood

During the experiments (see Section 5) we found that performance of the algorithm can significantly depend on the size of the chosen neighbourhood and a different size is appropriate for different problem domains. Therefore, in terms of accuracy of the algorithm, it is important to find the optimal neighbourhood. In fact, it is possible to estimate both the optimal value k and the optimal radius R for $S(tst, k)$ and $B(tst, R)$ neighbourhood respectively.

Below we describe the algorithm for estimating the optimal value k for $S(tst, k)$ neighbourhood. It would be similar if the optimal value k for *k-NN* method was estimated. The leave-one-out method is

used on a training set to estimate the accuracy of the classifier for different values of k ($1 \leq k \leq k_{max}$) and the value k for which the estimation is the greatest is chosen. Applying it directly would require repeating leave-one-out estimation k_{max} times. However, we emulate this process in time comparable to the single leave-one-out test for k equal to the maximal possible value $k = k_{max}$. Below we present the algorithm that implements this idea.

Algorithm 4.2. $getClassificationVector(trn, k_{max})$

$NN =$ vector of k_{max} training examples $NN_1, \ldots, NN_{k_{max}}$
nearest to trn sorted according to the distance $\varrho(trn, \cdot)$
for each decision $v \in V_d$ $decStrength[v] = 0$
$currentDec =$ the most frequent decision in $trnSet$
for $k = 1, 2, ..., k_{max}$
    if $isConsistent(rule_{trn}(NN_k), NN)$ **then**
        $v = d(NN_k)$
        $decStrength[v] = decStrength[v] + 1$
        if $decStrength[v] > decStrength[currentDec]$ **then** $currentDec = v$
    $A[k] = currentDec$
return A

Algorithm 4.3. $findOptimalK(k_{max})$

for each $trn \in trnSet$ $A_{trn} = getClassificationVector(trn, k_{max})$
return $\arg\max_k |\{trn \in trnSet : d(trn) = A_{trn}[k]\}|$

For a test example tst the function $getClassificationVector(tst, k_{max})$ finds k_{max} examples nearest to the example tst on average in linear time and sorts them according to the distance $\varrho(tst, \cdot)$. Next it returns the vector of decisions that $RIONA$ classifier would return for successive values of k. Algorithm 4.3 calls this routine for every training object and then it selects the value k for which the global estimation of accuracy is maximal.

Note that by ignoring the consistency checking in the function $getClassificationVector(tst, k_{max})$ we obtain the k nearest neighbours algorithm with selection of the optimal k. We call this classification algorithm *ONN* and we used it in experiments for comparison with *RIONA* and other algorithms (see Section 5). A new test object tst is classified by *ONN* with the most frequent decision in the set $S(tst, k)$, where the number k is selected as in the algorithm described above.

In the case of $B(tst, R)$ neighbourhood we applied a similar idea. Instead of considering k_{max} successive values in the *for* loop, appropriate intervals for radius R were considered. The experiments presented in the next section show that $S(tst, k)$ neighbourhood has better performance in terms of accuracy (see Section 5).

Table 1. The average optimal k, the average accuracy (%) and the standard deviation for *RIONA* with the optimal k-best neighbourhood and the average accuracy (%) for the other systems: *RIA*, *ONN*, *3-NN*, *RIONA* with the optimal $B(tst, R)$ neighbourhood, *C5.0*, *DeEPs* and *DeEPsNN*. The superscripts denote the confidence levels: 5 is 99.9%, 4 is 99%, 3 is 97.5%, 2 is 95%, 1 is 90%, and 0 is below 90%. Plus indicates that the average accuracy of an algorithm is higher than in *RIONA* and minus otherwise

Domain (size, attr, classes)	k_{opt}	RIONA	RIA	ONN	3-NN	RIONA(B)	C5.0	DeEPs	DeEPsNN
australian (690, 14, 2)	41,2	86,1±0,4	$65,0^{-5}$	$85,7^{-2}$	$85,0^{-4}$	$85,7^{-2}$	85,9	84,9	**88,4**
breast (277, 9, 2)	77,9	73,4±1,0	**73,9**^{0}	$73,0^{0}$	$68,6^{-5}$	$73,6^{0}$	-	-	-
breast-wis (683, 9, 2)	3,0	97,0±0,3	$89,7^{-5}$	$97,0^{0}$	**97,1**^{0}	$96,1^{-5}$	95,4	96,4	96,3
bupa-liver (345, 6, 2)	40,6	**66,6±1,7**	$63,0^{-5}$	$64,1^{-4}$	$66,0^{0}$	$66,4^{0}$	-	-	-
census (45222, 16, 2)	42,1	83,8±0,0	-	$84,1^{+5}$	$82,0^{-5}$	$83,9^{+5}$	85,8	**85,9**	**85,9**
chess (3196, 36, 2)	11,9	98,0±0,1	-	$96,9^{-5}$	$97,0^{-5}$	$97,5^{-5}$	**99,4**	97,8	97,8
german (1000, 20, 2)	29,2	**74,5±0,5**	$70,1^{-5}$	$74,1^{-1}$	$72,1^{-5}$	$73,1^{-4}$	71,3	74,4	74,4
glass (214, 9, 6)	2,1	70,7±1,9	$39,5^{-5}$	$70,7^{0}$	**71,9**$^{+1}$	$63,9^{-5}$	70,0	58,5	68,0
heart (270, 13, 2)	19,4	83,2±1,0	$62,8^{-5}$	$83,1^{0}$	$81,3^{-5}$	**83,4**^{0}	77,1	81,1	81,1
iris (150, 4, 3)	37,1	94,6±0,6	$90,5^{-5}$	$94,4^{0}$	$95,3^{+4}$	$94,7^{0}$	94,0	**96,0**	**96,0**
letter (20000, 16, 26)	3,8	**95,8±0,1**	-	**95,8**^{0}	**95,8**^{0}	$94,0^{-5}$	88,1	93,6	95,5
lymph (148, 18, 4)	1,4	85,4±1,3	$76,4^{-5}$	**86,3**$^{+1}$	$84,4^{-2}$	$81,4^{-5}$	74,9	75,4	84,1
mushroom (8124, 22, 2)	1,0	**100,0±0,0**	-	**100,0**^{0}	**100,0**^{0}	**100,0**^{0}	**100,0**	**100,0**	**100,0**
nursery (12960, 8, 5)	43,3	**99,3±0,0**	-	**99,3**^{0}	$98,1^{-5}$	$99,2^{-4}$	97,1	99,0	99,0
pendigits (10992, 16, 10)	1,2	**99,4±0,0**	-	**99,4**^{0}	**99,4**^{0}	$97,4^{-5}$	96,7	98,2	98,8
pima (768, 8, 2)	34,3	74,7±0,9	$65,2^{-5}$	$74,4^{0}$	$72,2^{-5}$	$72,7^{-5}$	73,0	**76,8**	73,2
primary (336, 15, 21)	75,9	31,7±0,8	$32,4^{+1}$	**40,3**$^{+5}$	$33,5^{+4}$	$31,6^{0}$	-	-	-
satimage (6435, 36, 6)	3,7	91,3±0,1	-	$91,3^{0}$	**91,4**$^{+2}$	$87,7^{-5}$	86,7	88,5	90,8
segment (2310, 19, 7)	1,7	97,4±0,1	$45,3^{-5}$	**97,5**$^{+2}$	$97,3^{-2}$	$92,1^{-5}$	97,3	95,0	96,6
shuttle (58000, 9, 7)	1,3	**99,9±0,0**	-	**99,9**^{0}	**99,9**^{0}	$99,8^{-5}$	99,6	97,0	99,7
solar-flare (1066, 10, 8)	70,9	81,2±0,3	$81,4^{+1}$	$82,7^{+5}$	$78,1^{-5}$	$81,7^{+5}$	82,7	**83,5**	**83,5**
splice (3186, 60, 3)	17,3	93,9±0,2	-	$93,9^{0}$	$94,0^{0}$	**94,6**$^{+5}$	94,2	69,7	69,7
wine (178, 13, 3)	10,1	**97,2±0,6**	$40,1^{-5}$	**97,2**^{0}	$96,9^{0}$	$94,5^{-5}$	93,3	95,6	95,5
yeast (1484, 8, 10)	23,0	**59,8±0,6**	$45,9^{-5}$	$58,1^{-5}$	$54,9^{-5}$	$59,1^{-4}$	56,1	**59,8**	54,6
Total Average[4]		**88,7±0,4**	64,3	**88,7**	87,8	87,3	86,6	86,1	87,1

5. Experimental Study

Table 1 presents experimental results for 24 data sets from the UCI repository [6]. For data that are split into a training and a testing set the experiments were performed for joined data (additionally the accuracy measured on test data is given in Section 5.2). We compare the performance of *RIONA* with both $S(tst, k)$ and $B(tst, R)$ neighbourhood to the performance of *RIA*, *ONN*, *3-NN*, *C5.0*, *DeEPs* and *DeEPsNN* systems. The accuracy for *C5.0*, *DeEPs* and *DeEPsNN* is taken from the paper [21]. The remaining algorithms were tested on a 800MHz PentiumIII PC, with 512M bytes of RAM. The algorithm *RIA* is time expensive so it was tested only for smaller data sets. The results were obtained by performing 10-fold cross-validation 10 times for each data set. All implemented algorithms: *RIONA, RIA, ONN, 3-NN* and *RIONA(B)* were tested with exactly the same folds and the significance of difference between algorithms was estimated using one-tailed paired t-test. The result of a single cross-validation test was the accuracy averaged over all 10 folds and the final average accuracy and the confidence level for difference between *RIONA* and the corresponding algorithm were computed from 10 repeats of the cross-validation test (for *census-income* and *shuttle* only 4 repeats). SVDM metric and the optimal neighbourhood were computed from a training set independently for each run in a cross-validation test.

For all data sets the presented results were obtained for the metric described in Section 3.2 and $NormNStrength$ measure for conflict resolving (see Section 4). Although, during preliminary experiments, we have also tried other types of metrics, no one appeared significantly better then the presented one in terms of accuracy on a range of problem domains. We have also compared both $NStrength$ and $NormNStrength$ measures as a strategy for conflict resolving and have obtained almost identical results. The optimal size of a neighbourhood was searched during the process of learning on the basis of the training examples. From the time complexity perspective it was important to limit searching for the optimal k to a small fixed range of possible values from 1 to k_{max} in such a way that sorting and consistency checking of k_{max} nearest neighbours were efficient. Since the values k_{max} optimal in this sense are the values close to the square root of the training set size (see Section 5.3) we set $k_{max} = 200$ (it is close to the square root of the size of the largest domains). In the next subsection we examine the significance of this setting. Before applying the algorithm no preprocessing was done. In particular we did not apply discretisation for numerical attributes.

In Table 1 it can be seen that the accuracy of *RIONA* and *ONN* is comparable or better than well-known classifiers, in particular, their accuracy is generally better than the accuracy of *RIA* and *3-NN*. It suggests the conclusion that *RIONA* and *ONN* may replace successfully both the rule-based algorithm using all minimal rules and the *k-NN* with a fixed *k*. It also proves that using a properly selected subset of rules in rule-based systems gives better results than using all minimal rules. The range of tested data sets indicates that the presented algorithms work well for domains with both numerical and symbolic attributes. In particular, it works well for numerical attributes without preprocessing.

5.1. RIONA versus ONN

While comparing *RIONA* and *ONN* (*k-NN* with selection of the optimal neighbourhood) it can be seen in Table 1 that significant differences in accuracy occurred mostly for smaller data sets (*breast, bupa-liver, chess, primary, solar-flare* and *yeast*). Differences for all other domains are less than 1 percent. The only difference between *RIONA* and *ONN* is the operation of consistency checking. In order to explain the similarity of results we checked what part of the k-neighbourhood for the optimal k is eliminated by the operation of consistency-checking (see Table 2).

The presented results show that only for three domains: *breast-cancer, primary* and *solar-flare* the operation of consistency checking eliminates a significant fraction of nearest neighbours. For other domains the number of consistent objects from the optimal neighbourhood in *RIONA* algorithm is close to the number of all objects from the optimal neighbourhood of *k-NN* algorithm. Therefore the differences in classification accuracy are small. These observations suggest that the operation of consistency checking in the algorithm *RIONA* is not very significant (see Section 4).

We suspect that this observations relates to the fact that usually the set of all consistent, minimal rules is of a large size. The last column in Table 2 indicates that the support set induced from the whole set of consistent and minimal rules contains a large fraction of all examples. On the other hand, the analysis of accuracy in dependence on a number of neighbours k shows that usually a small number of objects gives the best accuracy. It suggests that many of consistent and minimal rules are induced rather accidentally. Hence considering either a reasonably computed smaller set of rules or a more restrictive operation of consistency checking may give better classification accuracy.

Table 2. The dependence of a number of consistent objects on a number of nearest neighbours: the second column presents an optimal value k, the third column presents the average number of objects in the optimal k-neighbourhood that are consistent with a tested object and the fourth and fifth columns present analogical data for maximal experimented neighbourhood.

Domain	optimal $k = k_{opt}$	number of consistent for k_{opt}	maximal calculated $k = k_{max}$	number of consistent for k_{max}
australian	372	338,22	690	552,42
breast-cancer	123	19,17	275	22,82
census-income	256	192,01	501	344,85
chess	23	21,85	3196	794,05
german	34	33,32	1000	731,77
heart	15	14,96	270	241,44
iris	55	47,21	149	66,63
letter	3	2,99	758	443,72
lymphography	2	1,96	148	68,45
mushroom	1	1,0	508	477,97
nursery	590	377,81	1028	592,41
pendigits	1	1,0	514	498,7
primary	49	6,11	334	10,42
satimage	4	4,0	500	498,7
segment	1	1,0	2310	1716,29
shuttle	1	1,0	504	492,43
solar-flare	28	3,76	929	8,33
splice (dna)	15	14,93	2000	1835,73
yeast	21	20,28	1484	582,45

5.2. Further Study

In this section we describe some of the experiments and conclusions that led us to the final version of the presented algorithm. We also present experiments that can help us understand important aspects of the *RIONA*.

First, we performed the experiments that helped us compare two types of a neighbourhood: the radial neighbourhood $B(tst, R)$, where the radius R was variable and the k-best neighbourhood $S(tst, k)$, where k was variable, and choose a proper range of values of a variable parameter to test. Looking at the second and the third columns in Table 3 it can be seen that the accuracy of the algorithm for the neighbourhood $S(tst, k)$ is better than $B(tst, R)$ on 13 domains and worse only on 4 domains. For *breast-cancer* it lost about 1 percent, for *solar-flare* and *splice* it lost about half percent and for *census-income* the difference in accuracy is insignificant. On the other hand for a number of domains (*letter, lymphography, pendigits, satimage, segment, yeast*) the neighbourhood $S(tst, k)$ outperformed $B(tst, R)$ with the gain much higher than 1 percent of accuracy. Thus we can draw a conclusion that for the algorithm *RIONA* the neighbourhood $S(tst, k)$ works better than $B(tst, R)$, although the latter neighbourhood seems to be more natural. In further experiments we focused our attention on the neighbourhood $S(tst, k)$.

Table 3. Leave-one-out accuracy of *RIONA* on training sets for different types of a neighbourhood: the optimal $S(tst, k_{opt})$ (optimal k is given in parenthesis), the optimal $B(tst, R_{opt})$ (average k for optimal R) and the maximal experimented $S(tst, k_{max})$ (*all* denotes the unlimited neighbourhood). The last column contains the loss in accuracy (in comparison to the second column) when the maximal experimented value k was limited to $k_{max} = 200$.

Domain	RIONA for S(tst,k)	RIONA for B(tst,r)	RIONA for $S_{\triangle}$(tst,k) with maximal k	RIONA difference for $S_{\triangle}$(tst,k) with $k_{max} = 200$
australian	87,68% (372)	87,39% (381)	68,55% (all)	-0,15% (77)
breast-cancer	74,72% (123)	75,81% (125)	74,0% (all)	0,0%
census-income	83,97% (256)	84,02% (420)	83,81% (500)	-0,04% (24)
chess	98,31% (23)	98,15% (2176)	98,09 (all)	0,0%
german	76,5% (34)	75,7% (13)	70,1% (all)	0,0%
heart	85,55% (15)	85,18% (35)	64,07% (all)	0,0%
iris	96,0% (55)	96,0% (37)	91,33% (all)	0,0%
letter	95,38% (3)	93,39% (34)	74,69% (500)	0,0%
lymphography	87,83% (2)	84,45% (12)	81,08% (all)	0,0%
mushroom	100,0% (1)	100,0% (109)	99,97% (500)	0,0%
nursery	99,46% (590)	99,43% (674)	99,31% (1000)	-0,1% (171)
pendigits	99,53% (1)	97,37% (117)	82,21% (500)	0,0%
primary	32,44% (49)	32,12% (56)	31,84% (all)	0,0%
satimage	91,09% (4)	87,82% (231)	81,83% (500)	0,0%
segment	97,66% (1)	92,64% (90)	41,47% (all)	0,0%
shuttle	99,94% (1)	99,83% (195)	99,64% (500)	0,0%
solar-flare	81,42% (28)	81,98% (18)	81,42% (all)	0,0%
splice (dna)	94,9% (15)	95,55% (14)	62,5% (all)	0,0%
yeast	61,38% (21)	59,70% (115)	47,1% (all)	0,0%

The setting $k_{max} = 200$ preserved the efficiency of *RIONA* but the interesting question was how significantly this setting influenced the classification results. Please note that the maximal possible value k is just the size of a training set. In order to answer this question the following experiment was performed: for the smaller sets (less than 4000 objects) experiments were performed for all possible values of k and for the greater sets the maximal value k was set to $k_{max} = 500$ (for the set *nursery* we made the exception $k_{max} = 1000$). The classification accuracy was measured for the leave-one-out method applied to the whole set. Figures 1, 2, 3, 4 present the dependence of classification accuracy on the value of k for exemplary domains.

For most data sets we observed that while increasing k beyond a certain small value the classification accuracy was falling down (see Figures 1, 2, 3). In particular, while comparing the fourth and the second column in Table 3 one can see that for most data sets the results for the total or a quite large neighbourhood are significantly worse than the results for the neighbourhood found by the algorithm *RIONA*. For the remaining data sets (*breast, census-income, nursery, primary, solar-flare*) the accuracy becomes stable beyond a certain value k (see Figure 4).

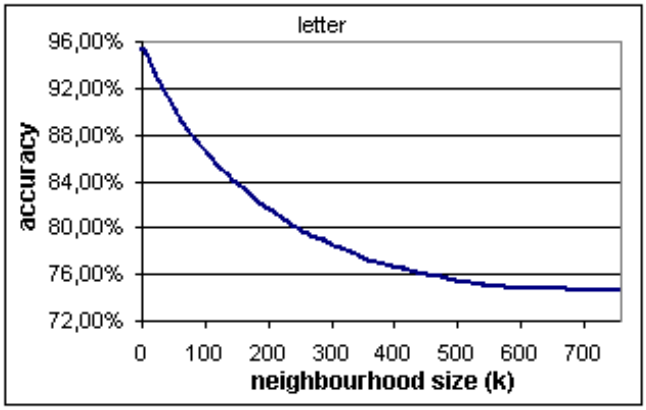

Figure 1. Classification accuracy for *letter*.

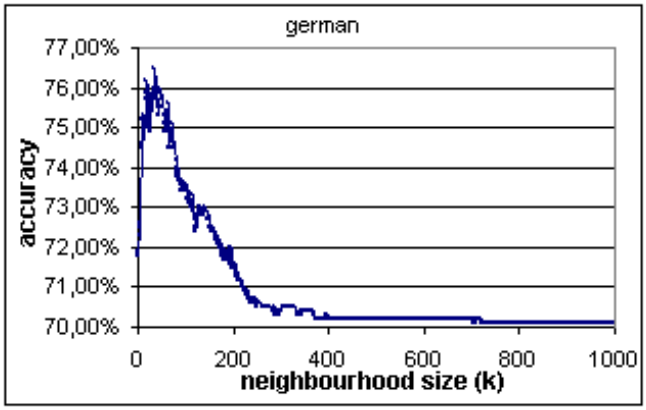

Figure 2. Classification accuracy for *german*

For the former group we examined the neighbourhood size (value of k) for which the maximum accuracy was obtained. In the latter case we examined both the value of k beyond which accuracy remains stable and the fluctuations in accuracy while increasing k. In the second column of Table 3 it can be seen that in most cases the optimal value of k is less than 200. Moreover, for many domains the optimal value of k is less than 60 and for 7 of them this value is equal or less than 4. On the other hand, for the domains where the optimal k was greater than 200 (*australian, census-income* and *nursery*) the loss in classification accuracy related to this setting was insignificant: it remained within the range of 0,15% (see the last column in Table 3). Moreover, the accuracy became stable for values of k also much lower than 200. Therefore we could conclude that the setting $k_{max} = 200$ preserved good time complexity properties and did not change the results significantly for tested data sets.

The fact that a small neighbourhood gives the best accuracy leads to another conclusion. Limiting the support set of a maximally general decision rule from $MinRules$ to a neighbourhood of a test example can be seen as replacing the rule with a more specific one. In this sense the presented results suggest that

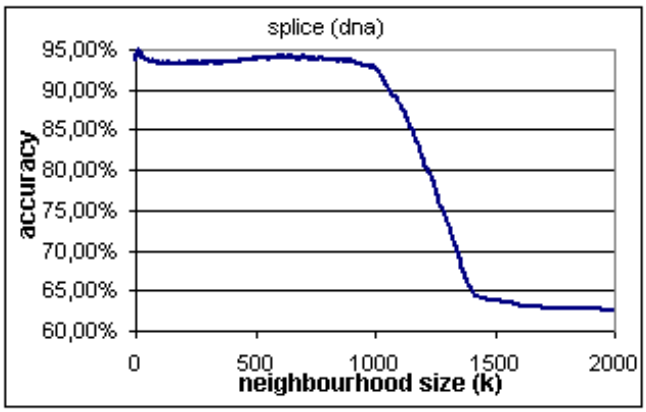

Figure 3. Classification accuracy for *splice*.

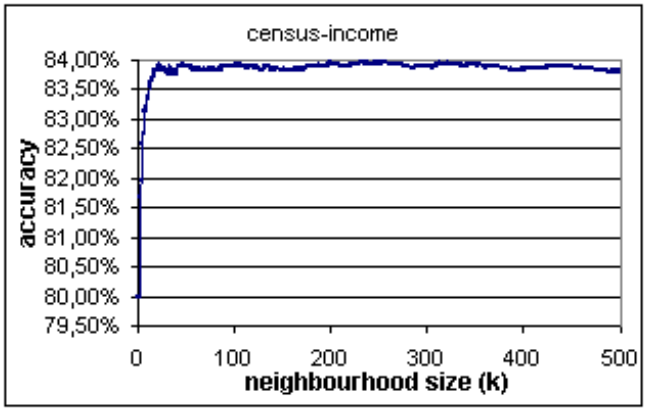

Figure 4. Classification accuracy for *census-income*.

taking the complete set of consistent and maximally general decision rules usually gives worse accuracy than a set of more specific rules. It relates to measures for conflict resolving that consider as one of the important factor the specificity of a rule (see e.g. [17]). It would be interesting to do further research in order to verify this hypothesis.

For a number of data sets (*letter, pendigits, satimage, segment, shuttle* and *yeast*) we noticed that the accuracy is falling down monotonically. Since for these domains the best accuracy is obtained for the smallest values of k, *k-NN* method seems to work best for them. On the other hand, all the mentioned data have numerical attributes. Hence, we can conclude that numerical data are induced best by the *k-NN* method and a falling accuracy characterises well the data sets that are appropriate for *k-NN* method.

Analogical experiments were done for the neighbourhood $B(tst, R)$. The conclusion was similar to the one just mentioned. For all the presented domains we observed that after the value R exceeded some constant R_{max} (where R_{max} was relatively small in comparison to all possible values of R) classification accuracy either became worse or did not improve significantly. This suggests the similar conclusion, i.e. the best accuracy is obtained for small radius.

Table 4. Difference in classification accuracy between induced and optimal number of different neighbours: the second column presents the accuracy on a test set for the value k induced from a training set, the third column presents maximal possible accuracy of the algorithm if the optimal number k were found (the optimal number k is given in parentheses), the last column shows the difference between these two accuracies.

Domain	RIONA for S_ϱ(tst,k)	optimal accuracy for S_ϱ(tst,k)	accuracy difference
census-income	83,74% (256)	83,75% (417)	-0,01%
letter	95,54% (3)	95,74% (7)	-0,2%
pendigits	97,36% (1)	97,88% (4)	-0,52%
satimage	91,1% (4)	91,1% (3)	0,0%
shuttle	99,94% (1)	99,94% (1)	0,0%
splice (dna)	94,18% (15)	94,26% (33)	-0,08%

If data are split into training and testing set one can ask the question whether the accuracy on a test set obtained for the value k computed from a training set may differ significantly from the optimal accuracy on a test set. In order to study this aspect we compared this accuracies on the data sets that were originally split. The results are presented in Table 4. The experiments showed that for *pendigits* accuracy obtained by *RIONA* differs by about half percent from the accuracy with the optimal number k and for other domains the difference remains in the range of 0.2%. It means that the used algorithm finds almost the optimal number k in terms of the accuracy obtained.

To sum up, there is no need to take the whole training set in the process of classification. Moreover, taking less objects can improve classification performance. It was found that the performance is significantly better for the $S(tst, k)$ neighbourhood in comparison to the $B(tst, R)$ neighbourhood.

5.3. Time Complexity of RIONA

First, the learning algorithm performs two phases for each training object. In the first phase it selects k_{max} nearest objects among n objects, where n denotes the size of a training set. On average it is done in the linear time. In the second phase the algorithm sorts all k_{max} selected objects and checks consistency among them. It takes $O(k_{max}^2)$. Finally, for the whole training set the algorithm computes leave-one-out accuracy for each $1 \leq k \leq k_{max}$. It takes $O(nk_{max})$. Summing up, the average complexity of the learning algorithm is $O(n(n + k_{max}^2))$. In practice the component $O(n^2)$ is predominant.

Testing is analogical to learning. The classification algorithm finds k_{opt} nearest examples and then checks consistency among them. Since $k_{opt} \leq k_{max}$, the complexity is $O(n + k_{max}^2)$ for a single test object and the total average complexity of the testing algorithm is $O(m(n+k_{max}^2))$ where m is a number of test objects. In Table 5 one can see that for all the presented data sets the average time of classification for a single object is less than 0.6 s. Moreover, for larger data sets it is comparable with a single object test time in the algorithm *ONN* and is much shorter than a single test object time in the algorithm *RIA*.

In case when the number of test objects is approximately equal to the number of training objects, taking into account both the learning and the classification phase, the average time complexity of *RIONA* is in practise $O(n^2)$, while the average time complexity of *RIA* is $O(n^3)$ what is quite a significant acceleration.

Table 5. Single object test time (in seconds) for *RIONA*, *RIA* and *ONN*

Domain	t_{RIONA}	t_{RIA}	t_{ONN}	Domain	t_{RIONA}	t_{RIA}	t_{ONN}
australian	0,026	0,087	0,022	breast	0,016	0,021	0,014
breast-wis	0,032	0,063	0,017	bupa-liver	0,009	0,016	0,006
census	0,572	$> 5,0$	0,568	chess	0,130	0,891	0,126
german	0,047	0,188	0,042	glass	0,010	0,012	0,006
heart	0,019	0,024	0,014	iris	0,003	0,006	0,003
letter	0,236	$> 5,0$	0,224	lymph	0,017	0,019	0,014
mushroom	0,223	$> 5,0$	0,219	nursery	0,169	$> 5,0$	0,167
pendigits	0,133	$> 5,0$	0,130	pima	0,013	0,055	0,010
primary-tumor	0,018	0,028	0,018	satimage	0,174	$> 5,0$	0,169
segment	0,046	0,557	0,042	shuttle	0,378	$> 5,0$	0,376
solar-flare	0,025	0,082	0,023	splice	0,405	3,194	0,393
wine	0,010	0,891	0,007	yeast	0,017	0,104	0,014

6. Conclusions and Future Research

The research reported in the paper attempts to bring together the features of rule induction and instance-based learning in a single algorithm.

As the empirical results indicate the presented algorithm obtained the accuracy comparable to the well-known systems: *3-NN*, *C5.0*, *DeEPs* and *DeEPsNN*. The experiments show that the choice of a metric is very important for classification accuracy of the algorithm. The combination of the normalised Manhattan metric for numerical attributes and SVDM metric for symbolic attributes proved to be very successful. It did not require discretisation for numerical attributes.

We have compared two types of a neighbourhood: the k-nearest neighbours ($S(tst, k)$) and the ball $B(tst, R)$. The former type of a neighbourhood gave generally better results, although the latter seemed more natural. This may suggest that the topology of the space induced by the used metric is rather complex.

We found that the appropriate choice of the neighbourhood size is also an important factor for classification accuracy. It appeared that for all domain problems the optimal accuracy is obtained for a small neighbourhood (a small number of nearest neighbours k in S or a small radius R in B neighbourhood). This leads us to the conclusion that generally it is enough to consider only a small neighbourhood instead of the maximal neighbourhood related to the whole training set. This is interesting from the classification perspective, because it suggest that usually only a small number of training examples is relevant for accurate classification. We believe that this fact can stimulate further research in this direction. It also illustrates the empirical fact that while using rule-based classifiers one can obtain better results by rejecting some rules instead of using all minimal rules like the algorithm *RIA* does. We propose an approach to use only the rules that are built on the basis of a neighbourhood of the test case.

The fact mentioned above is also the key idea that allowed us to make the original algorithm *RIA* (see Section 3.1) efficient without loss in classification accuracy. In practice the complexity of learning and classification is only squarely and linearly dependent on the size of a learning sample respectively. Although a great effort was put into accelerating the algorithm, we think that further acceleration is

possible, for instance by more specialised data structures and an approximate choice of nearest examples (see e.g. [22], [25]).

The facts that *RIONA* and *ONN* algorithms have similar classification accuracy and the fraction of objects eliminated by the consistency checking operation is very small indicate that this operation has rather small influence on the accuracy of the presented algorithm. It suggests that the *k-NN* component remains a dominant element of the presented algorithm and shows that either the construction of local rules should be more general or the operation of consistency checking should be more restrictive.

In *RIONA* the selection of the optimal value of k is performed globally. One possible extension of this approach is to apply a local method to searching for the appropriate value of k (see e.g. [31]).

The interesting topic, although marginal in this paper, is the dependence of the average number of training examples on the distance to a test case. Empirically it was noticed that the dependence was close to linear, what seemed interesting and surprising to us.

Acknowledgements

The authors are very grateful to professor Andrzej Skowron, dr Marcin Szczuka, dr Dominik Ślęzak, mgr Piotr Synak, and dr Jakub Wróblewski for useful remarks on this presentation. This work was supported by the grants 8 T11C 009 19 and 8 T11C 025 19 from the Polish National Committee for Scientific Research.

References

[1] Aha, D.W., Kibler, D. and Albert, M.K. (1991). *Instance-based learning algorithms*. Machine Learning, 6, pages 37-66.

[2] Bazan, J.G. (1998). *Discovery of decision rules by matching new objects against data tables.* In: L. Polkowski, A. Skowron (eds.), Proceedings of the First International Conference on Rough Sets and Current Trends in Computing (RSCTC-98), Warsaw, Poland, pages 521-528.

[3] Bazan, J.G., Szczuka M. (2000). *RSES and RSESlib - A Collection of Tools for Rough Set Computations.* In: W. Ziarko, Y. Yao (eds.), Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing (RSCTC-2000), Banff, Canada, pages 106-113.

[4] Biberman, Y. (1994). *A context similarity measure.* Proceedings of the Ninth European Conference on Machine Learning, Springer-Verlag, Catania, Italy, pages 49-63.

[5] Bishop, C.M. (1996). *Neural networks for pattern recognition*. Oxford University Press, Oxford, England.

[6] Blake, C.L., Merz, C.J. (1998). *UCI Repository of machine learning databases* [http://www.ics.uci.edu /˜mlearn/MLRepository.html], Department of Information and Computer Science, University of California, Irvine, CA.

[7] Clark, P. and Niblett, T. (1989). *The CN2 induction algorithm*. Machine Learning, 3, pages 261–284.

[8] Cost, S. and Salzberg, S. (1993). *A weighted nearest neighbor algorithm for learning with symbolic features.* Machine Learning, 10, pages 57-78.

[9] Cover, T.M. and Hart, P.E. (1967). *Nearest neighbor pattern classification.* IEEE Transactions on Information Theory, 13, pages 21-27.

[10] Domingos, P. (1996). *Unifying instance-based and rule-based induction.* Machine Learning, 24(2), pages 141-168.

[11] Duda, R.O. and Hart, P.E. (1973). *Pattern classification and scene analysis.* Wiley, New York, NY.

[12] Friedman, J.H., Kohavi, R., Yun, Y. (1996). *Lazy Decision Trees.* Proceedings of the Thirteenth National Conference on Artificial Intelligence, Cambridge, pp. 717-724, MA: MIT Press.

[13] Golding, A.R., Rosenbloom, P.S. (1991). *Improving rule-based systems through case-based reasoning*. Proceedings of AAAI-91, Anaheim, CA, pages 22-27.

[14] Góra, G., Wojna. A., (2002). *RIONA: A Classifier Combining Rule Induction and k-NN Method with Automated Selection of Optimal Neighbourhood.* In: T. Elomaa, H. Mannila, H. T. T. Toivonen (eds.), Proceedings of the Thirteenth European Conference on Machine Learning (ECML-2002), Helsinki, Finland.

[15] Góra, G., Wojna. A., (2002). *Local Attribute Value Grouping for Lazy Rule Induction.* In: J. Peters, A. Skowron, N. Zhong (eds.), Proceedings of the Third International Conference on Rough Sets and Current Trends in Computing (RSCTC-2002), Penn State Great Valley, PA.

[16] GrzymalaBusse, J.W. (1992). *LERS—A system for learning from examples based on rough sets.* In: R. Slowinski (Ed.) Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory. Kluwer Academic Publishers, Dordrecht, Boston, London, pp. 3–18.

[17] Grzymala-Busse, J.W. (1998). *Applications of the Rule Induction System LERS.* In: Polkowski, L., Skowron A. (eds.) Rough sets in knowledge discovery 1 - methodology and applications, Physica-Verlag, Heidelberg, pages 366-375.

[18] Grzymala-Busse, J.W., Stefanowski, J. (1997). *Discretization of numerical attributes by direct use of the LEM2 induction algorithm with interval extension.* Proceedings of the VI Int. Symp. on Intelligent Information Systems, Zakopane, IPI PAN Press, pp. 149-158.

[19] Li, J., Dong, G., Ramamohanarao, K. (2000). *Instance-based classification by emerging patterns*. Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases, Springer-Verlag, Lyon, France, pages 191-200.

[20] Li, J., Ramamohanarao, K. and Dong, G. (2001). *Combining the strength of pattern frequency and distance for classification*. The Fifth Pacific-Asia Conference On Knowledge Discovery and Data Mining, Hong Kong, pages 455-466.

[21] Li, J., Dong, G., Ramamohanaro, K. and Wong, L. (2001). *DeEPs: A new instance-based discovery and classification system.* [http://sdmc.krdl.org.sg:8080/~limsoon/limsoonpapers.html], School of Computing, National University of Singapore.

[22] Maneewongvatana, S. and Mount, D.M. (2001). *On the Efficiency of Nearest Neighbor Searching with Data Clustered in Lower Dimensions*. In: V.N. Alexandrov, J. Dongarra, B.A. Juliano, R.S. Renner, Ch.J.K. Tan (eds.), Proceedings of the International Conference on Computer Science (ICCS-2001), San Francisco, USA, pages 842-851.

[23] Michalski, R.S. (1983). *A theory and methodology of inductive learning.* Artificial Intelligence, 20, pages 111-161.

[24] Michalski, R.S., Mozetic, I., Hong, J. and Lavrac, N. (1986) *The Multi-Purpose Incremental Learning System AQ15 and its Testing to Three Medical Domains*. Proceedings of AAAI-86, Morgan Kaufmann, San Mateo, pages 1041-1045.

[25] Mitchell T.M. (1997). *Machine learning*. McGraw-Hill, Portland.

[26] Nguyen, H. Son and Skowron, A. (1995). *Quantization of real value attributes,* Proceedings of the 2nd Annual Join Conference on Information Sciences, Wrightsville Beach, NC, pages 34-37.

[27] Quinlan, J.R. (1993). *C4.5: Programs for machine learning*, Morgan Kaufmann, San Mateo, California.

[28] Schaffer, C. (1994). A conservation law for generalisation performance. Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann, New Brunswick, NJ, pages 259-265.

[29] Skowron, A. and Rauszer, C. (1992). *The Discernibility Matrices and Functions in Information Systems*. R. Słowiński (ed.), Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory, Kluwer, Dordrecht, pages 331-362.

[30] Stanfill, C. and Waltz, D. (1986). *Toward memory-based reasoning.* Communications of the ACM, 29, pages 1213-1228.

[31] Wettschereck, D. (1994). *A study of Distance-Based Machine Learning Algorithms*. Doctor of Philosophy dissertation in Computer Science, Oregon State University.