Noname manuscript No.
(will be inserted by the editor)

Representative-based classification through covering-based neighborhood rough sets

Ben-Wen Zhang · Fan Min(✉) · Davide Ciucci

May 6, 2015

Abstract Considerable progress has been made in the theory of covering-based rough sets. However, there has been a lack of research on their application to classification tasks, especially for nominal data. In this paper, we propose a representative-based classification approach for nominal data using covering-based rough sets. The classifier training task considers three issues. First, we define the neighborhood of an instance. The size of the neighborhood is determined by a similarity threshold θ. Second, we determine the maximal neighborhood of each instance in the positive region by computing its minimal θ value. These neighborhoods form a covering of the positive region. Third, we employ two covering reduction techniques to select a minimal set of instances called *representatives*. To classify a new instance, we compute its similarity with each representative. The similarity and minimal θ of the representative determine the distance. Representatives with the minimal distance are employed to obtain the class label. Experimental results on different datasets indicate that the classifier is comparable with or better than the ID3, C4.5, NEC, and NCR algorithms.

Keywords. classifier, covering-based rough set, neighborhood, representative, similarity

B.-W. Zhang
School of Computer Science, Southwest Petroleum University, Chengdu 610500, China
Department of Computer Science, Sichuan University for Nationalities, Kangding 626001, China
e-mail: zhbwin@163.com

F. Min(✉)
School of Computer Science, Southwest Petroleum University, Chengdu 610500, China
e-mail: minfanphd@163.com

D. Ciucci
DISCo, University of Milano-Bicocca, viale Sarca 336/14, 20126 Milano, Italy
e-mail: ciucci@disco.unimib.it

1 Introduction

There are two general classification approaches in data mining [1]. One is model-based learning, such as decision trees [2,3], rule sets [4,5,6], neural networks [7], and support vector machines (SVM) [8,9]. These approaches build a classifier on the training set, and then classify new instances using this classifier while ignoring the training set. The advantages of model-based learning include the low storage requirements of the model and the low time complexity of classification. The second classification approach is instance-based lazy learning, such as k-nearest neighbors (kNN) [10,11]. This lazy learning has no training stage. A new instance is simply compared to existing ones, with the k-nearest neighbors being selected for classification. An advantage of this approach is that it naturally deals with changes in the data.

Covering-based rough sets, first introduced by Zakowski [12], have attracted considerable research interest in recent years [13]. Yao et al. proposed a framework for the study of covering-based rough set approximations [14], whereas Zhu et al. explored the topological properties of these sets [15] and studied their connections with relation-based rough sets [16,17]. Wang et al. introduced matroid theory to covering-based rough sets [18], and proposed three types of existing covering approximation operators using Boolean matrices [19]. Hu et al. developed a soft fuzzy rough set model to reduce the influence of noise [20], an attribute reduction algorithm based on a generalized fuzzy-rough model to efficiently remove redundant attributes [21], and a neighborhood rough set model to classify numeric data [22]. Du et al. extended the formal theoretical framework to numerical feature spaces, and derived a neighborhood covering reduction-based approach to extract rules from numeric data [23]. Qian et al. proposed reduction algorithms using three parallelism strategies in cloud computing. However, there has been a lack of research into the application of covering-based rough sets for classifying nominal data.

In this paper, we propose a representative-based classification approach that uses covering-based neighborhood rough sets. There are three issues for the classifier training task. First, we define the neighborhood of an instance by considering the similarity between instances. The size of the neighborhood is determined by a similarity threshold θ. Second, we determine the maximal neighborhood of each instance in the positive region. This is done by choosing a minimal θ such that the whole neighborhood is within the positive region. These neighborhoods, also called blocks, form a covering of the positive region. Third, we employ two covering reduction techniques to select a minimal subset of these blocks. One removes blocks that are a subset of another block, and the other greedily selects the block covering the largest number of uncovered instances. Instances forming these blocks are called *representatives*. To classify a new instance, we compute its similarity with each representative [24]. This similarity and the minimal θ of the representative determine the *relative distance*. Representatives with the minimal relative distance are used to obtain the class label. Our approach is essentially an integration of both model-based and lazy learning techniques. In the training stage, representatives are selected and their neighborhood thresholds computed. From this viewpoint, it is a model-based learning approach. In the testing stage, the distances between a new instance and existing representatives are computed. From this viewpoint, it is a lazy learning approach.

Generally, our approach is similar to those described by Hu et al. [25] and Du et al. [23], which both employ covering-based neighborhood rough sets. One major difference is that these previous studies deal with numeric data, whereas we consider nominal data. Consequently, to define the neighborhood, we use a *similarity* measure instead of a distance measure. Moreover, in the testing stage, we do not use blocks formed in the training stage, as these blocks may not cover some new instances. Instead, we define a new distance measure by considering the similarity measure and threshold learning in the training set. A new instance is assigned the class label of the nearest representatives. In this way, all new instances can be classified. Simple voting [26] is employed for conflict resolution, because the importance of each representative has been considered through the distance measure.

Experiments are undertaken on 10 UCI datasets [27], including Tic-tac-toe, Zoo, Voting, Mushroom, and so on. The results show that our approach is more accurate on Voting and Mushroom. For these two datasets, the number of representatives is smaller than the size of the dataset. For the Mushroom dataset, the ratio is about 2.5%. In other words, the average number of instances covered by the neighborhood of each representative is around 40. This phenomenon indicates that our approach is most appropriate for data with low representative ratios. A comparison study shows that the precision of our approach is often better than that of the ID3 algorithm [28] and comparable to that of the C4.5 algorithm [29, 30]. Our approach outperforms the NEC [25] and NCR [23] algorithms in terms of precision for nominal data.

The remainder of this paper is organized as follows. Section 2 introduces some preliminary knowledge and proposes new definitions for the similarity and neighborhoods. We consider the problem decomposition in Section 3. Section 4 discusses a representative generation algorithm and a representative-based classification algorithm. In Section 5, we present experimental results that illustrate the classification precision of our model. Finally, we state some conclusions from this research, and suggest further study ideas, in Section 6.

2 Preliminaries

In this section, we review some basic concepts related to decision systems, indiscernibility relations and similarity relations, neighborhoods, and covering. Furthermore, we introduce the notion of a minimal threshold that will be used to select representative elements in the proposed approach.

2.1 Decision system

The concept of a decision system [31,32,33] is widely used in data mining [34] and machine learning.

Definition 1 [33] A decision system S is a 3-tuple:

$$S = (U, C, D), \tag{1}$$

where U is a finite set of instances called the universe, C is the set of conditional attributes, and D is the set of decision attributes.

In this paper, we only consider a symbolic decision system with one decision attribute. This is denoted as $S = (U, C, \{d\})$, where $C = \{a_1, \ldots, a_i\}$ and d is the decision attribute.

Table 1 An example of a decision table.

U	Headache	Temperature	Lymphocyte	Leukocyte	Eosinophil	Flu
x_1	Yes	High	High	Low	Normal	Yes
x_2	No	Normal	Low	High	Low	Yes
x_3	No	Normal	Low	High	Low	Yes
x_4	Yes	High	High	High	Normal	Yes
x_5	Yes	High	High	High	High	Yes
x_6	Yes	High	High	Low	High	Yes
x_7	Yes	High	Normal	Normal	High	No
x_8	No	Low	Normal	Normal	High	No
x_9	Yes	Low	Normal	Normal	High	No
x_{10}	Yes	Normal	High	Low	High	No
x_{11}	Yes	High	High	Low	High	No

For example, Table 1 lists a decision system, where U=$\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}\}$, C = {Headache, Temperature, Lymphocyte, Leukocyte, Eosinophil}, and d = {Flu}.

2.2 Indiscernibility and Similarity Relations

In a decision system, instances in U can be divided into several subsets according to arbitrary attributes. Each subset of attributes determines a classification of all instances into classes having the same description in terms of these attributes.

For example, in Table 1, we can divide the universe into two parts X_1, X_2 using the decision attribute: $U/\{d\} = \{X_1, X_2\}$, where $X_1 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $X_2 = \{x_7, x_8, x_9, x_{10}, x_{11}\}$.

In a decision system, we can describe the relationship between instances through their conditional attribute values.

Definition 2 [35] Let S be a decision system. Then, $\forall A \subseteq C$, $\forall x, y \in U$, the indiscernibility relation $IND(A)$ is defined as:

$$IND(A) = \{(x, y) \in U \times U : a(x) = a(y), \forall a \in A\}. \tag{2}$$

It is easy to see that the indiscernibility relation defined in this way is an equivalence relation on U that partitions the universe U into equivalence classes: $[x]_A = \{y \in U | (x, y) \in IND(A)\}$. Using these equivalence classes, we are able to define the lower and upper approximation of any subset of the universe $X \subseteq U$ as:

$$l_A(X) = \{x \in U | [x]_A \subseteq X\}, \tag{3}$$

$$u_A(X) = \{x \in U | [x]_A \cap X \neq \emptyset\}. \tag{4}$$

The lower approximation $l_A(X)$ can be interpreted as the elements *definitely* belonging to X, whereas the upper approximation considers those *possibly* belonging to X. Now, let S be a decision system and X_i its decision classes. If two instances

have the same conditional attribute values but different decision attribute values, they are called contradictory. Decision systems with contradictory instances are *inconsistent*. There are certain techniques, including those based on rough sets [36,37,38], that deal with such inconsistency. However, in this work, we will not consider such a situation. Therefore, we will remove these contradictory instances from U to form a new universe. Indeed, using the lower approximation, we can point out the elements that can be classified with certainty given our knowledge (represented by a set of attributes A). This collection of elements is named the *positive region*:

$$POS_A(d) = \bigcup_{X_i \in U/\{d\}} l_A(X_i). \tag{5}$$

We can observe that two objects in U fall into $IND(C)$ if and only if they have the same values on *all* attributes in C according to Definition 2. Such a case may be too strict to be used in many applications. To overcome this issue, several generalized models have been defined [13,39]. First, by relaxing the transitivity requirement of the indiscernibility relation, we obtain a *similarity* relation. In this case, we do not get a partition, but a covering. That is, the similarity classes are not necessarily disjoint. In the case of numerical attributes, similarity is usually defined using a distance. In the case of categorical data, there are several possible ways of defining similarity relations, usually based on a similarity measure. For an overview, we refer to [40]. In our work, we consider the *overlap* measure and the induced similarity.

Definition 3 Let $S = (U, C, \{d\})$ be a categorical decision system. The *overlap measure* between $x, y \in U$ with respect to $A \subseteq C$ is:

$$sim(x, y, A) = \frac{|\{a \in A | a(x) = a(y)\}|}{|A|}, \tag{6}$$

and the *overlap similarity* $OS(x, y, A) = \{(x, y) | sim(x, y, A) \geq \theta\}$ for a given threshold $\theta \in \{\frac{1}{|A|}, \frac{2}{|A|}, \ldots, 1\}$.

Note that this concept is the same as the measure for the *quantitative indiscernibility relation* discussed in [35]. Thus, the overlap measure counts the percentage of equal attributes between two objects, and if this percentage is higher than a fixed threshold, the two objects are said to be similar. For example, in Table 1, because none of the conditional attribute values are shared by x_1 and x_2, we obtain $sim(x_1, x_2) = \frac{0}{5} = 0$. Hence, regardless of the threshold, x_1 and x_2 are not similar. On the other hand, x_1 and x_4 have four conditional attribute values in common, so $sim(x_1, x_4) = \frac{4}{5} = 0.8$ and x_1 and x_4 are similar for any $\theta \geq 0.8$.

A similarity relation is the first step towards (at least) two different generalizations of rough sets: relation- and covering-based. We now briefly introduce some concepts of covering rough sets.

2.3 Neighborhood and Covering Rough Sets

At the heart of standard rough set theory is the notion of a partition, where equivalence classes cover the universe and are disjoint. If we remove this second requirement, we obtain a covering.

Definition 4 Given a universe U, a *covering* of U is a collection of sets $C_i \subseteq \mathcal{P}(U)$ such that $\cup C_i = U$.

As a partition is obtained by an equivalence relation, a covering can be obtained by clustering objects according to a similarity relation.

Definition 5 The *neighborhood* of $x \in U$ with respect to a similarity measure $Sm : U \times U \mapsto \mathcal{R}$ and a threshold $\theta \in \mathcal{R}$ is:

$$nh(x, \theta) = \{y \in U | Sm(x, y) \geq \theta\}. \tag{7}$$

Here, the threshold θ is given by the users. Obviously, $nh(x, \theta)$ gradually increases as the threshold θ decreases. For example, considering the overlap measure in Table 1, we get $U/\{d\} = \{X_1, X_2\}$, where $X_1 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $X_2 = \{x_7, x_8, x_9, x_{10}, x_{11}\}$. As the similarity degree θ decreases, we can respectively obtain $nh(x_1, 1) = \{x_1\}$, $nh(x_1, 0.8) = \{x_1, x_4, x_6, x_{11}\}$, and $n(x_1, 0.6) = \{x_1, x_4, x_5, x_6, x_{10}, x_{11}\}$.

In the extreme case, θ is equal to zero and all instances are neighbors of the given one. We are more interested in the minimal possible value of θ.

Definition 6 Let $S = (U, C, \{d\})$ be a categorical decision system, $A \subseteq C$, and $U/\{d\} = \{X_1, X_2, \ldots, X_k\}$. The minimal possible threshold value for $x \in X_i$ is:

$$\theta_x^* = \min\{\theta | nh(x, \theta) \subseteq X_i\}. \tag{8}$$

Note that θ_x^* is not specified by the user. Rather, it is determined by the decision system and x. This determines, in turn, the maximal possible neighborhood of x.

Definition 7 The maximal neighborhood of $x \in U$ is:

$$nh^*(x) = nh(x, \theta_x^*). \tag{9}$$

Fig. 1 illustrates the determination of $\theta_{x_1}^*$ with overlap similarity. Here, we observe $nh(x_4, 1) = \{x_4\}$, $nh(x_4, 0.8) = \{x_1, x_4, x_5\}$, and $nh(x_4, 0.6) = \{x_1, x_4, x_5, x_6, x_{11}\}$. Since $d(x_1) = d(x_5) = d(x_4)$, $\theta_{x_1}^* \leq 0.8$. However, $d(x_{11}) \neq d(x_4)$, and hence $\theta_{x_4}^* > 0.6$. Consequently, $0.6 < \theta_{x_4}^* \leq 0.8$, and $nh^*(x_4) = \{x_1, x_4, x_5\}$. For simplicity, we let $\theta_{x_1}^* = 0.8$.

Finally, one may wonder whether a covering can be somehow simplified. Indeed, as the elements of the covering can overlap, a set C_i can be the union of some others, or it can be a subset of another and so on. According to the application, one may wish to avoid some of these situations containing redundant information. In this sense, several notions of covering *reduction* have been proposed in the literature [41,42]. Here, we are interested in maximizing the size of the neighborhoods, thus eliminating small (redundant) ones.

Definition 8 Let $\mathcal{C} = \{C_i\}$ be a covering of a set U. A set $C_i \in \mathcal{C}$ is *redundant* if it is a subset of another set $C_j \in \mathcal{C}$. A *reduct* of $\mathcal{C}$ is the covering obtained by eliminating redundant sets from $\mathcal{C}$.

A covering not containing redundant elements is also called a *genuine* covering [43,44].

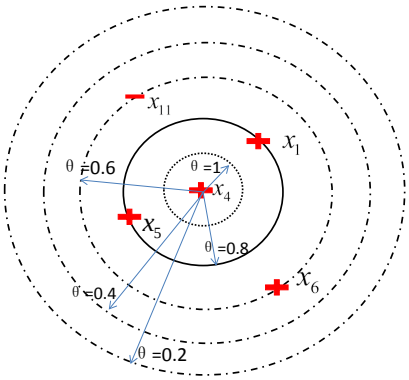


Fig. 1 An illustration of θ_x^*.

3 Problem decomposition

In this section, we consider the general classification problem. Both conditional attributes and decisional attributes are nominal. There is no additional information such as the weight of each attribute, test cost of data [45,46,47,48], or misclassification costs for wrong predictions [49,50].

Problem 1 Classification
Input: The training set TR and the test set TS.
Output: Prediction of the decision attribute values of the test set.
Optimization object: Maximize the precision.

Many popular classifiers use the same technique for training and testing. For example, the ID3 algorithm trains a decision tree using the training set. This decision tree is employed for classification on the test set.

Here, we propose a new framework for classification. The key issue is to decompose Problem 1 into two subproblems, one for training and one for testing. These problems are given by Problems 2 and 3, respectively.

Problem 2 Representative selection
Input: The training set TR.
Output: A representative set $Y \subseteq TR$ and respective θ_x^* for any $x \in Y$.
Constraint: $CR = \{(x, \theta_x^*) | x \in Y\}$ is a covering of TR, namely $\cup CR = TR$.
Optimization object: Minimize $|Y|$.

Problem 3 Representative-based prediction
Input: The test set TS, representative instances set Y, and respective θ_x^* for any $x \in Y$.
Output: Predicted class label of any $x' \in TS$.
Optimization objective: Maximize the precision.

There are a number of issues concerning Problem 2. First, similar to feature selection, representative selection is essential in reducing the data size. In fact, there are other popular representative selection techniques such as SVM [8,51], which

are mainly designed for numeric data. Second, as a representative, each selected instance should preserve the information of a group of instances. This group is given by the neighborhood of the representative. Third, the set of representatives as a whole should preserve certain properties of the data. As indicated by the constraint, all instances in the training set (or, more accurately, all instances in the positive region) should be covered by the neighborhoods of one or more representatives. Fourth, each representative and respective neighborhood essentially correspond to a decision rule in the training set. The coverage of the neighborhoods can also be viewed as the coverage of the decision rule set. However, this rule set is not necessarily employed for classification on the test set. This issue will be discussed later. Fifth, as indicated by the optimization objective, the number of representatives should be as small as possible. According to the principle of Occam's razor [52], smaller rule sets are more reliable.

There are also a number of issues concerning Problem 3. First, the input of the problem includes the output of Problem 2, namely the representatives and their respective thresholds. In other words, many instances in the training set are not considered in this problem. Second, it is easy to predict instances that fall within the control range of some representative neighborhoods, as each representative naturally corresponds to a decision rule. However, for instances beyond the control of all representatives, prediction is not straightforward if we employ rule-based approaches. Third, as indicated by the optimization objective, we maximize the prediction precision.

4 Proposed algorithms

In this section, we propose two algorithms to deal with Problems 2 and 3. These are the representative generation algorithm and the representative-based classification algorithm, respectively.

4.1 The representative generation algorithm

In this subsection, we describe the representative generation algorithm. There are two stages:

1. Neighborhood construction. Based on Definition 7, the maximal neighborhood of each instance is constructed.
2. Representative selection and redundancy removal. A greedy approach is designed to select a set of representatives whose neighborhoods cover the training set. Another approach is designed to remove redundant representatives introduced in the process of greedy selection.

Algorithm 1 lists the representative generation algorithm. Steps 1–3 correspond to neighborhood construction, and Step 4 corresponds to representative selection and redundancy removal. Specifically, Step 1 computes the similarity among all instance pairs in the training set. Step 2 computes the threshold θ_x^* for each instance, and Step 3 constructs the neighborhood of each instance. Step 4.1 selects representatives through covering block selection, and Step 4.2 removes redundant representatives.

Algorithm 1 Representative generation RG

Input: The decision system $S = (U, C, \{d\})$.
Output: A representative instances set Y and a covering $CR = \{(x, \theta_x^*)|x \in Y\}$.
Constraint: $Y \subseteq U$ and $\bigcup CR = POS_C(d)$.
Optimization objective: Minimize $|Y|$.

1: $Y = \emptyset, CR = \emptyset$; // initialize the output.
//Step 1: Compute all instances similarity in U.
2: Compute $sim(x, y)$, where $(x, y) \in U \times U$;
//Step 2: Compute θ_x^* in U.
3: **for** (each $x \in U$) **do**
4: low = 0, upper = 1, middleTheta = (low + upper)/2;
5: **while** (upper - low)$\geq \frac{1}{|C|}$ **do**
6: **for** (each $y \in U$) **do**
7: **if** ($sim(x, y) \geq$ middleTheta) $\wedge (d(x) \neq d(y))$ **then**
8: low = middTheta;
9: break;
10: **end if**
11: **if** ($y = |U| - 1$) **then**
12: upper = middTheta;
13: **end if**
14: **end for**
15: **end while**
16: $\theta_x^* = middleTheta$;
17: **end for**
//Step 3: Construct neighborhood $nh^*(x)$
18: **for** (each $x \in U$) **do**
19: $nh^*(x) = \emptyset$;
20: **for** ($y \in U$) **do**
21: **if** ($sim(x, y) \geq \theta_x^*$) **then**
22: $nh^*(x) = \cup\{y\}$;
23: **end if**
24: **end for**
25: **end for**
//Step 4: Representative selection and removal.
26: Compute $U/\{d\} = \{X_1, X_2, \ldots, X_{|v_d|}\}$;
27: **for** ($i = 1$ to $|v_d|$) **do**
28: $X = X_i$;
//Step 4.1: Selection.
29: **while** $X \neq \emptyset$ **do**
30: Select $x \in U \cap X_i$ st. $|nh^*(x) \cap X|$ is maximal;
31: $Y_i = Y_i \cup \{x\}$;
32: $X = X - nh^*(x)$;
33: **end while**
//Step 4.2: Removal.
34: **for** (each $x \subseteq CR_i$) **do**
35: **if** ($\bigcup_{x' \in CR_i - \{x\}} nh^*(x') = X_i$) **then**
36: $Y_i = Y_i - \{x\}$;
37: **end if**
38: **end for**
39: **end for**
40: $CR = \{(x, \theta_x^*)|x \in Y\}$;
41: Return Y and CR.

We describe the algorithm in further detail through an example. Step 1 corresponds to Lines 1–2 of the algorithm. We compute the similarity between instance pairs in the training set according to Equation (6). The similarity is then used

in Steps 2 and 3 to calculate the maximal neighborhood of an instance. Table 2 presents the similarity between the instances in Table 1. Naturally, $sim(x, y) = sim(y, x)$ and $sim(x, x) = 1$. Therefore, we can ignore approximately half of the elements.

Table 2 Similarity between instance pairs.

x_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
x_1	1	0	0	0.8	0.6	0.8	0.4	0	0.2	0.6	0.8
x_2	-	1	1	0.2	0.2	0	0	0.2	0	0.2	0
x_3	-	-	1	0.2	0.2	0	0	0.2	0	0.2	0
x_4	-	-	-	1	0.8	0.6	0.4	0	0.2	0.4	0.6
x_5	-	-	-	-	1	0.8	0.6	0.2	0.4	0.6	0.8
x_4	-	-	-	-	-	1	0.6	0.2	0.4	0.8	1
x_7	-	-	-	-	-	-	1	0.6	0.8	0.4	0.6
x_8	-	-	-	-	-	-	-	1	0.8	0.2	0.2
x_9	-	-	-	-	-	-	-	-	1	0.4	0.4
x_{10}	-	-	-	-	-	-	-	-	-	1	0.8
x_{11}	-	-	-	-	-	-	-	-	-	-	1

Step 2 corresponds to Lines 3–17 of the algorithm. We compute θ_x^* using a binary search for each instance according to Definition 6. This takes advantage of Table 2 and the decision attributes of Table 1. Any instance whose similarity with x is greater than or equal to θ_x^* should be subjected to the same decision as x. Since there are m attributes, there are $m + 1$ possible values of θ_x^*, i.e., 0, $\frac{1}{m}$, $\frac{2}{m}$, ..., and 1. This process is illustrated in Fig. 1.

Step 3 corresponds to Lines 18–28. We obtain the neighborhood according to θ_x^* computed in Step 2. Table 3 lists the maximal neighborhood and respective θ^* of each instance. With only five conditional attributes, there are six possible similarity thresholds, i.e., 0, $\frac{1}{5}$, $\frac{2}{5}$, ..., and 1. We observe that $\theta_{x_i}^*$ is never equal to any threshold. This is because we use a binary search to compute θ_x^*. In this way, $\theta_x^* = 0.88$ is equivalent to $\theta_x^* = 1$, and $\theta_x^* = 0.12$ is equivalent to $\theta_x^* = 0.2$.

Table 3 Neighborhoods and θ^*.

x_i	$nh^*(x_i)$	$\theta_{x_i}^*$
x_1	$\{x_1\}$	0.88
x_2	$\{x_2, x_3, x_4, x_5\}$	0.12
x_3	$\{x_2, x_3, x_4, x_5\}$	0.12
x_4	$\{x_1, x_4, x_5\}$	0.62
x_5	$\{x_5\}$	0.88
x_6	$\{x_6\}$	0.88
x_7	$\{x_7, x_9\}$	0.62
x_8	$\{x_7, x_8, x_9, x_{10}, x_{11}\}$	0.12
x_9	$\{x_7, x_8, x_9, x_{10}, x_{11}\}$	0.38
x_{10}	$\{x_{10}\}$	0.88
x_{11}	$\{x_{11}\}$	0.88

Step 4 corresponds to Lines 26–41. It follows the neighborhood construction method. Step 4.1 selects some blocks to cover $POC_C(\{d\})$. To minimize the number of neighborhoods, we use a greedy strategy. Each time we choose the largest block

that could cover the remaining part of $POC_C(\{d\})$. Line 30 allows us to choose the block that covers the largest part of X. Although some blocks may be larger, the areas of X they cover are smaller. These are therefore not selected. The main reason for this situation is that some blocks are interleaved sets. We select blocks until $POC_C(\{d\})$ has been completely covered. Step 4.2 then removes redundant blocks (if any). Redundant blocks may occasionally appear as a result of the greedy selection strategy. Although not redundant when selected, they may be covered by other blocks that are subsequently added. Therefore, some removal process is required.

We now analyze the time complexity of Algorithm 1.

Proposition 1 *The complexity of Algorithm 1 is $O(n^2(p+m))$, where n is the number of objects, m is the number of attributes, and p is the number of representatives.*

Proof First, we compute the similarity of all instances in U, as described in Step 1. Let m be the number of attributes and n be the number of instances in the training set. We must compare the current instance with all other instances on all of the attributes. That is, every instance needs to be compared with another $(n-1)$ instances on each attribute. Hence, the time cost of comparing and judging is $(n-1) \times m$ for one instance, and so the time complexity of this step is $O(n^2 m)$.

Second, we compute the maximal threshold θ^*, as described in Step 2.We use a binary search to obtain θ_x^*. As there are m attributes, there are $m+1$ possible values of θ_x^* (i.e., 0, $\frac{1}{m}$, $\frac{2}{m}$, ..., and 1). We need to compute the value of $\lceil \log m \rceil$ n times with respect to the other $n-1$ instances to determine θ^*. The time cost of computing θ^* is $n \times (n-1) \times \lceil \log m \rceil$, giving a time complexity for this step of $O(n^2 \log m)$.

Third, we also need to obtain the covering block for each instance, as described in Step 3. If $sim(x, y) \geq \theta_x^*$, instance y belongs to $nh^*(x)$. As we need to compare the threshold θ^* with all instance similarities, the time cost in computing the covering block for each instance is $(n-1)$. The total time cost for this step is $n \times (n-1)$, and so the time complexity of this step is $O(n^2)$.

Finally, we must select representatives using covering reduction and remove redundant blocks, as described in Step 4. We employ a greedy strategy to select representatives that cover maximal instances. Each time a representative is selected, we recalculate the number of instances that are covered by blocks. Thus, we have n comparisons for each block. Let p be the number of representative selected. The time cost will be $n \times (n-1) \times p$, giving a time complexity in Step 4.1 of $O(n^2 p)$. To remove redundant blocks, we must determine whether each block is completely covered by other blocks. The time cost is $p \times (p-1) \times n$, and so the time complexity in Step 4.2 is $O(p^2 n)$. The number of representatives p is much smaller than the number of instances n. Hence, the time complexity in this step is $O(n^2 p)$.

To summarize, let n be the number of instances, m be the number of attributes in the training set, and p be the number of representatives. Since $p \leq n$, the time complexity of Algorithm 1 is:

$$O(n^2 m) + O(n^2 \log m) + O(n^2) + O(n^2 p) = O(n^2(p+m)). \tag{10}$$

4.2 The representative-based classification algorithm

In this subsection, we describe the representative-based classification algorithm. One approach is to view each neighborhood of the input as a decision rule. A new instance falling into a neighborhood will be assigned the class label of the respective representative. However, for many new instances, there may be no neighborhood that covers it. Therefore, we borrow an idea from the kNN method to design our algorithm.

A key issue of kNN-like algorithms is how to define or design the distance function. Let Y be the set of representatives. The distance between instance x' and a representative $x \in Y$ is:

$$distance(x', x) = \frac{1}{sim(x', x)} - \frac{1}{\theta_x^*}. \tag{11}$$

This distance function considers not only the similarity of instances, but also the minimal neighborhood threshold, which is an input of the problem. Note that the distance cannot simply be interpreted as that in a Euclidean space. It may be negative or zero, which means that x' is in the range of $nh^*(x)$. If the distance is positive, then x' is beyond the control range. In Fig. 2, we can observe that $distance(x'_2, x_2) < 0$ and $distance(x'_2, x_3) < 0$, whereas $distance(x'_2, x_1) > 0$. In particular, $distance(x'_1, x_1) = distance(x'_1, x_2) = distance(x'_1, x_3) = 0$.

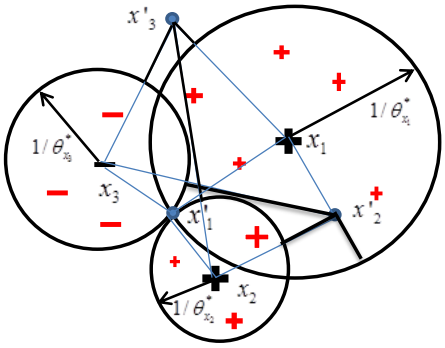


Fig. 2 The distance between test instances and representatives.

We now discuss how to determine the class label of a new instance x'. We calculate the distance between x' and each $x \in Y$. The set of representatives that are the minimal distance from x' is:

$$X = \{x \in Y | distance(x', x) = mds(x', Y)\}, \tag{12}$$

where

$$mds(x', Y) = \min\{distance(x', x) | x \in Y\}. \tag{13}$$

Algorithm 2 Representative-based classification *(RC)*

Input: A test case x', representative instances set Y, and covering $CR = \{(x, \theta_x^*)|x \in Y\}$.
Output: Predicted class label of x'.

1: $DS = 0$; //store the current distance.
2: MDS = MAX_VALUE; // store the minimum distance.
3: $X = \emptyset$; //store representatives for which $distance(x', x) = MDS$.
4: **for** (each $x \in Y$) **do**
5: Compute $sim(x', x)$;
6: Compute distance $DS = distance(x', x)$ according to Equation (11);
//seek the minimum distance.
7: **if** $(DS < MDS)$ **then**
8: $MDS = DS$;
9: $X = \{x\}$;
10: **else** $\{(DS = MDS)\}$
11: $X = X \cup \{x\}$;
12: **end if**
13: **end for**
//Predicted class label of x'.
14: Compute $d'(x')$ according to Equation (14);
15: Return $d'(x')$.

Equation (13) gives the minimal distance between x' and Y, which we denote as MDS. The predicted class of x' is:

$$d'(x') = \arg \max_{1 \le i \le |v_d|} |\{x \in X | d(x) = i\}|. \tag{14}$$

In other words, we use the standard voting method according to Equation (14). In Fig. 2, it is easy to see that the decision attribute value of x_2' is $'+'$. The minimal distance between instance x_1' and all representatives is zero. Hence, the decision attribute value of x_1' is $'+'$, because there exist two representatives whose attribute values are $'+'$ and only one representative that is $'-'$.

Algorithm 2 gives the pseudo-code for the representative-based classification algorithm. This uses Equations (11) and (14). Note that, in Line 9, X is reset to contain only one representative. Line 11 adds a new representative x to X while $distance(x', x) = MDS$.

In Algorithm 2, we need to compare each test instance with all representatives. We use all attributes to compute the similarity between a test instance and a representative. The time cost of computing the similarity between test instances and all representatives is $p \times m$, where p is the number of representatives and m is the number of attributes. The time complexities of computing the distance and seeking the minimal distance are $O(p)$. Overall, the time complexity of Algorithm 2 is $O(pm)$.

5 Experiments

In this section, we describe the results of extensive computational tests that essentially address the following questions.

1. What kind of datasets is our algorithm most appropriate for?
2. How does the size of the training set influence the performance of our approach?

3. How does the performance of the proposed approach compare with that of algorithms such as ID3, C4.5, NEC, and NCR?

To compare with other classifiers, we must consider suitable evaluation measures. For existing approaches such as ID3, some new instances may not be classified. Therefore, we must consider not only precision, but also recall. We will adopt the widely used F-measure [53,54] for this purpose. Let P be the proportion of correctly classified instances with respect to all classified instances, and R be the proportion of correctly classified instances with respect to all instances in the test set. The F-measure is given by

$$F_1 = \frac{2PR}{P+R}. \tag{15}$$

The most important parameter in the NEC algorithm is the threshold δ, which determines the size of the neighborhood. Hu et al. [25] described a method to compute this threshold as $\delta = min(\Delta(x_i, s)) + w \cdot range(\Delta(x_i, s)), w \leq 1$, where x_i is the set of training objects, and $min(\Delta(x_i, s))$ is the minimal distance between x_i and the test object s. The $range(\Delta(x_i, s))$ is the value range of $\Delta(x_i, s)$ with respect to all training objects. In this case, the threshold δ is dynamically assigned based on the training objects and test objects. More important is how to determine the value of the parameter w. To compare with the NEC algorithm, we use an empirical range of w.

5.1 Experimental setup

Experiments were undertaken on 10 real-world datasets from the UCI machine learning repository. These include numeric datasets, such as Wpbc, Wdbc, Wine, and Iris, which were discretized on all attributes. The others contain nominal or categorical data. A description of these datasets is given in Table 4.

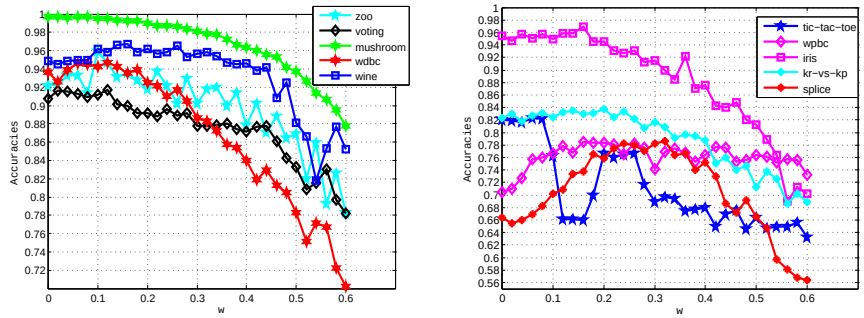
The general experimental scenario was repeated for each dataset. We randomly divided each dataset into two parts, one for training and the other for testing. To obtain training and test sets of different scales, we used six different proportions to divide the datasets. The Mushroom dataset contains a large number of instances, and there are more attributes contained in the Splice and Kr-vs-kp datasets. Hence, we varied the proportions for these datasets from 0.02 to 0.12 in steps of 0.02. The proportion of training data in the other datasets was varied from 0.1 to 0.6 in steps of 0.1. We compare the accuracy of our RC algorithm with that of ID3 and C4.5 implemented in WEKA (where C4.5 is also called J48). We also present a comparison with the NEC and NCR algorithms. We undertook 10 experiments for each division, and computed the average accuracy.

To conduct the experiments, it was necessary to determine the optimal value of w, which controls the size of the neighborhood in the method of [25]. We increased w from 0 to 0.6 in steps of 0.02, and computed the classification accuracies on the 50% split for all datasets except *Mushroom*, *Kr-vs-kp*, and *Splice*. The percentage split for these three datasets was only 10%. Fig. 3 (a), (b) show the classification accuracy with respect to w for all datasets. We can see a similar trend in each curve, with the accuracy first increasing and then decreasing as the threshold w rises. The optimal accuracies occur near the point $w = 0.1$; hence, we set the threshold to $w = 0.1$ in the NEC algorithm.

Table 4 Description of four datasets.

	Attributes	Class	Instances
Iris	5	3	150
Tic-tac-toe	10	2	958
Wine	14	3	178
Voting	17	2	435
Zoo	17	8	101
Mushroom	23	2	8124
Wdbc	31	2	569
Wpbc	34	2	198
Kr-vs-kp	37	2	3196
Splice	61	3	3190

The NCR algorithm greedily searches the largest neighborhood of instances in the forward search step, and removes the redundant rules. If the margin of training instances is less than zero, we set the margins to zero. This indicates that this instance covers itself while its margin is less than or equal to zero. If no rule matched the test object, it was classified to the class of the nearest neighborhood.



(a) (b)

Fig. 3 Classification accuracy with respect to w.

5.2 Results

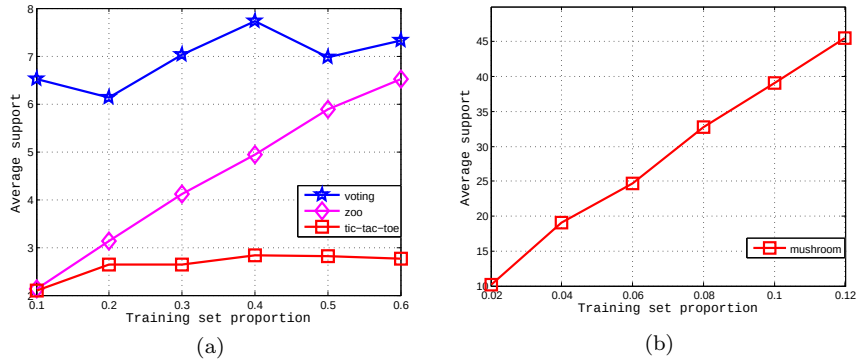
Table 5 lists the average number of representatives in some of the datasets. We can see that the number of representatives increases as the scale increases. The rate of increase is relatively slow for the Zoo and Mushroom datasets. For example, for the Voting dataset, the number of representatives increases linearly with respect to the number of training data, whereas the number of representatives for the Mushroom dataset only increases from 16 to 21.9 (nearly 40% increase) as the number of training data increases by a factor of five. The number of representatives can be viewed as the number of rules. Therefore, a slower increment indicates that our approach is more appropriate for the dataset.

Table 5 Average number of representatives for four datasets.

Training-scale	0.1	0.2	0.3	0.4	0.5	0.6
Tic-tac-toe	45.2	72.4	108.6	135.4	170.6	207.4
Voting	6.6	14.2	18.5	22.5	31.1	35.6
Zoo	4.7	6.4	7.3	8.1	8.5	9.2
Training-scale	0.02	0.04	0.06	0.08	0.1	0.12
Mushroom	16	17	19.7	19.8	20.8	21.9

Fig. 4 illustrates the average support of the representatives. The size of a block denotes the number of instances it contains. Because each representative corresponds to a block, the size of the block is called the *support* of the representative. The *average support* of all representatives indicates the degree of similarity among objects. Naturally, a larger support indicates more generality in the representative, and a better ability to predict new instances. The overlap among blocks means that the average support of the representatives times the number of representatives is greater than the size of the training set. Therefore, we need to study the average support of each representative independently.

From Fig. 4, we can observe that increasing the size of the training set produces a different increase in the average support of representatives for each dataset. The Tic-tac-toe and Voting datasets remain unchanged, indicating that new instances seldom fall into existing blocks. This suggests these datasets are more irregular. In contrast, the Mushroom and Zoo datasets display a linear increase, which implies that they are quite regular. These results indicate again that our approach is more appropriate for the Mushroom and Zoo datasets. Generally, both Table 5 and Fig. 4 reveal this phenomenon, even though there is no strict relationship between the average support of representatives, the number of representatives, and the size of the training set.



(a) (b)

Fig. 4 Average support of representatives for four datasets.

Fig. 5 shows the runtime and number of basic operations of Algorithm 1 for three bigger datasets. All the experiments were carried out on a personal computer

running Windows 8.1 with an Intel(R) Core(TM) i5-3210M CPU @2.50 GHz and 4.00 GB memory. We used the Java SE 1.7 programming language.

Fig. 5 (a), (c) shows the runtime and number of basic operations with respect to n. We can see that the results comply with our complexity analysis. On the Splice dataset with $n = 3,000$, there are some 7.5 billion basic operations taking about 22 s. Hence, the algorithm is efficient and appropriate for real applications.

Fig. 5 (b), (d) shows the runtime and number of basic operations with respect to m. We selected 1,000 instances from these three datasets, and varied the number of attributes from 2 to 21. Although these two subfigures exhibit a similar trend, the lines are not smooth. This is because the change in p with respect to the increase in m is somewhat irregular.

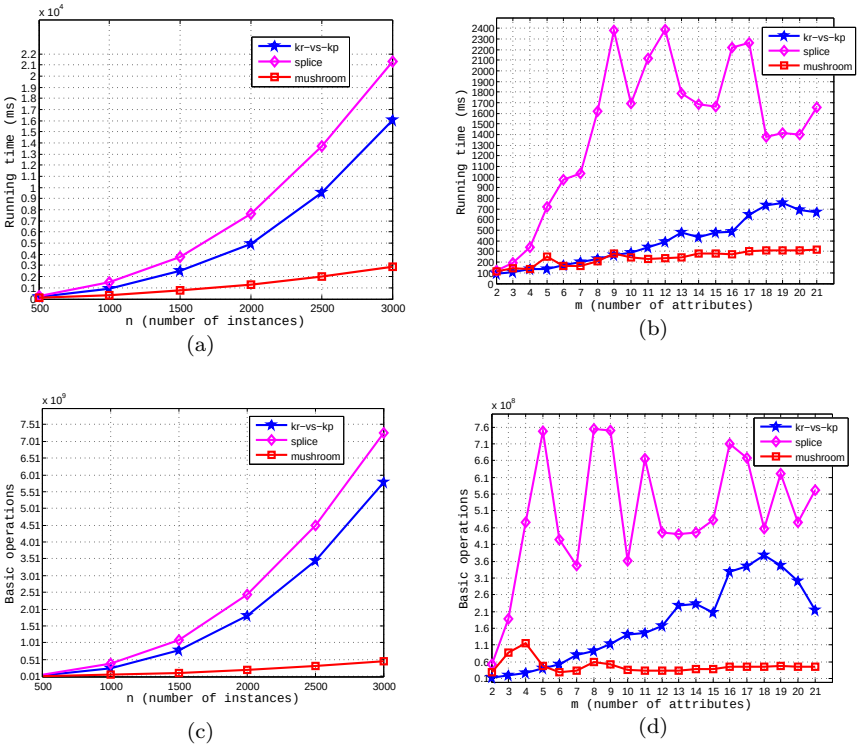


Fig. 5 Runtime and basic operations on three datasets.

Table 6 lists the average classification accuracy and standard deviation for each dataset. As the number of training instances increases, the classification accuracy improves. The classification accuracies are different for each dataset. For the first split percentage, all datasets produced poor classification accuracies. For example, the classification accuracies of *Iris*, *Wpbc*, and *Splice* are 0.6304, 0.6587 and 0.6504, respectively. However, the results become more reliable when the scale of the training set increases beyond 0.6, because the classification accuracies exceed 80%, except for dataset *Wpbc*.

The classification results are excellent for *Mushroom*, but not so good for *Iris*, *Wpbc*, and *Splice*. This is because the block size barely changes as n increases. The number of representatives increases rapidly, indicating that the number of rules also increases. This indicates that the similarity between these instances is lower. When the scale is very small, all datasets have very low classification accuracy, except for *Mushroom*. This is because the number of representatives remains largely unchanged with the increase in n. Hence, the support of the representatives increases quickly. Thus, the datasets most suited to our approach have smaller values of p compared with n and a bigger support for the representatives.

Table 6 Classification performance using the RC algorithm.

Training-scale	0.1	0.2	0.3	0.4	0.5	0.6
Iris	63.04 ± 8.51	71.75 ± 6.78	76.38 ± 5.21	79.11 ± 2.66	77.87 ± 6.63	81.68 ± 3.85
Tic-tac-toe	71.69 ± 1.81	77.34 ± 1.93	79.84 ± 13.9	81.60 ± 1.3	83.95 ± 1.26	86.22 ± 1.13
Wine	73.04 ± 2.79	83.08 ± 3.14	87.52 ± 3.54	87.48 ± 1.66	89.21 ± 2.71	91.94 ± 3.26
Voting	88.90 ± 2.57	92.13 ± 1.03	92.20 ± 0.98	92.15 ± 1.272	93.26 ± 1.51	93.51 ± 1.09
Zoo	76.81 ± 7.04	85.00 ± 5.41	90.99 ± 2.75	92.62 ± 3.88	92.62 ± 3.88	95.37 ± 3.14
Wdbc	91.19 ± 1.39	92.48 ± 1.25	92.78 ± 1.16	92.66 ± 1.30	94.21 ± 1.17	93.51 ± 0.80
Wpbc	65.87 ± 7.75	66.60 ± 2.91	67.77 ± 3.46	68.07 ± 5.00	71.01 ± 4.42	69.88 ± 4.58
Training-scale	0.02	0.04	0.06	0.08	0.1	0.12
Mushroom	97.56 ± 1.06	98.26 ± 0.55	99.21 ± 0.28	99.26 ± 0.27	99.57 ± 0.23	99.59 ± 0.21
Kr-vs-kp	76.01 ± 1.41	78.92 ± 2.21	82.34 ± 1.06	83.17 ± 1.09	84.79 ± 0.89	85.33 ± 1.22
Splice	65.04 ± 1.82	68.03 ± 2.25	70.25 ± 1.72	79.34 ± 1.23	80.11 ± 0.59	81.43 ± 0.67

Fig. 6 compares the RC, ID3, C4.5, NEC, and NCR algorithms in terms of the F-measure F_1 instead of the accuracy. ID3 cannot classify all instances, and the accuracy is computed as the number of correctly classified instances divided by the total number. Thus, the single accuracy may not provide a comprehensive measure.

The other four algorithms classify all instances in the test dataset. It is obvious that $R = P$ on this occasion, according to Equation (15), and we finally observe $F_1 = P$. F_1 coincides with the accuracy in this situation. With the ID3 algorithm, the unclassified instances generally cause R to be smaller than P. The value of F_1 increases with the scale of the training set on these datasets.

Fig. 6 compares the classification accuracies for different classifiers and percentage splits. Fig. 6 (a)–(f) shows the accuracy for nominal data, and the other subfigures indicate the accuracy with numerical data. Comparing these accuracies, we find that the classification performance of our approach is superior to that of the NEC and NCR algorithms for nominal data, but weaker than the NEC and NCR algorithms for numerical data. This suggests that our approach is better suited to nominal data. We can also see that our algorithm produced a better classification performance than the ID3 algorithm (Fig. 6 (a), (c), (g), (h), and (i)). The performance is comparable in Fig.6 (b), (d), (f), and (j), and slightly worse with the *Kr-vs-kp* dataset. Overall, this means that our approach is better than the ID3 algorithm. Fig. 6 (a), (c), (g), and (h) indicate that our approach is better than the C4.5 algorithm, whereas Fig. 6 (b), (e), (f), (i), and (j) suggest the opposite. Only on the *Mushroom* dataset is the performance comparable. From the above experimental analysis, we can state the following conclusions: our representative-based

neighborhood classifier is a simple, efficient, and powerful classification system, especially for nominal datasets.

5.3 Discussion

We can now answer the questions posed at the beginning of this section.

1. Our algorithms are appropriate for datasets in which the number of representatives is very small compared with the number of objects and the support of the representative blocks is large. This is because the bigger support indicates better generality. Table 5, Fig. 4, and Table 6 illustrate this phenomenon.
2. As Table 6 shows, classification quality improves as the scale of the training set increases. Therefore, our algorithm is more appropriate for larger datasets. The performance improves smoothly with the increase in training data density.
3. A comparison study showed that the performance of our approach is comparable to that of the classical ID3 and C4.5 algorithms. It is also better than the NEC and NCR algorithms for nominal data. Because NEC and NCR are naturally suited to numeric data, our algorithm performed worse with the discretized data. On the other hand, when we transferred nominal data to numeric data directly for NEC and NCR, they performed worse than our algorithm. This indicates that we should choose appropriate algorithms for different data types.

6 Conclusions and future work

In this paper, we have proposed a classification technique with two subalgorithms by integrating model-based and lazy learning techniques. A number of representatives are selected using a covering-based neighborhood rough set. These representatives are employed for classification through a kNN-like approach. This algorithm takes advantages of both types of learning techniques. Its performance is better than that of the NEC and NCR algorithms for nominal data.

In future work, we intend to compare, under our framework, similarity measures other than the overlap one studied here. Further, we plan to improve our method to handle inconsistent datasets. Indeed, we have shown that deleting certain objects can work effectively, but can cause some errors in classification. Finally, we would like to design more algorithms under this framework to deal with other classification problems, such as cost-sensitive scenarios.

Acknowledgements

This work is in part supported by the National Natural Science Foundation of China under Grant Nos. 61379089, 61379049 and Department of Education of Sichuan Province under Grant No. 13ZA0136.

References

1. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI magazine **17**(3) (1996) 37
2. Quinlan, J.: Induction of decision trees. Machine Learning **1** (1986) 81–106
3. Li, H.X., Zhou, X.Z.: Risk decision making based on decision-theoretic rough set: a three-way view decision model. International Journal of Computational Intelligence Systems **4**(1) (2011) 1–11
4. Homaifar, A., McCormick, E.: Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. Fuzzy Systems **3** (1995) 129–139
5. Li, H.X., Yao, Y.Y., Zhou, X.Z., Huang, B.: A two-phase model for learning rules from incomplete data. Fundamenta Informaticae **94** (2009) 219–232
6. Liu, D., Li, T.R., Li, H.X.: A multiple-category classification approach with decision-theoretic Rough sets. Fundamenta Informaticae **155**(2–3) (2012) 173–188
7. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2** (1989) 359–366
8. Joachims, T.: Making large scale svm learning practical. (1999)
9. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local svm approach. In: Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on. Volume 3., IEEE (2004) 32–36
10. Zhang, M.L., Zhou, Z.H.: Ml-knn: A lazy learning approach to multi-label learning. Pattern Recognition **40** (2007)
11. Atkeson, C.G., Moore, A.W., Schaal, S.: Locally weighted learning for control. Lazy Learning **54** (1997) 75–113
12. W. Zakowski: Approximations in the space (u, π). Demonstratio Mathematica **16**(40) (1983) 761–769
13. Yao, J., Ciucci, D., Zhang, Y. In: Generalized Rough Sets. Springer (2015)
14. Yao, Y.Y., Yao, B.X.: Covering based rough set approximations. Information Sciences **200** (2012) 91–107
15. Zhu, W.: Topological approaches to covering rough sets. Information Sciences **177**(6) (2007) 1499–1508
16. Zhu, W.: Relationship between generalized rough sets based on binary relation and covering. Information Sciences **179**(3) (2009) 210–225
17. Zhu, W.: Relationship among basic concepts in covering-based rough sets. Information Sciences **179**(14) (2009) 2478–2486
18. Wang, S.P., Zhu, W.: Matroidal structure of covering-based rough sets through the upper approximation number. International Journal of Granular Computing, Rough Sets and Intelligent Systems **2** (2011) 141–148
19. Wang, S.P., Zhu, W., Zhu, Q.X., Min, F.: Characteristic matrix of covering and its application to boolean matrix decomposition. Information Sciences **263** (2014) 186–197
20. Hu, Q.H., An, S., Yu, D.R.: Soft fuzzy rough sets for robust feature evaluation and selection. Information Sciences **180** (2010) 4384–4400
21. Hu, Q.H., Xie, Z.X., Yu, D.R.: Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation. Pattern Recognition **40** (2007) 3509–3521
22. Hu, Q.H., Yu, D.R., Liu, J.F., Wu, C.X.: Neighborhood rough set based heterogeneous feature subset selection. Information Sciences **178** (2008) 3577–3594
23. Du, Y., Hu, Q., Zhu, P., Ma, P.: Rule learning for classification based on neighborhood covering reduction. Information Sciences **181**(24) (2011) 5457–5467
24. Sun, L., Xu, J.C., Tian, Y.: Feature selection using rough entropy-based uncertainty measures in incomplete decision systems. Knowledge-Based Systems **36** (2012) 206–216
25. Hu, Q.H., Yu, D.R., Xie, Z.X.: Neighborhood classifiers. Expert Systems with Applications **34** (2008) 866–876
26. Satterthwaite, M.A.: Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. Journal of Economic Theory **10** (1975) 187–217
27. Blake, C.L., Merz, C.J.: UCI repository of machine learning databasesn. http://www.ics.uci.edu/~{}mlearn/MLRepository.html (1998)
28. Utgoff, P.E.: Id: An incremental id3. (1987)
29. Quinlan, J.R.: C4. 5: programs for machine learning. Volume 1. Morgan kaufmann (1993)
30. Quinlan, J.R.: Bagging, boosting, and c4. 5. In: AAAI/IAAI, Vol. 1. (1996) 725–730

31. Peterson, R., Silver, E.A.: Decision systems for inventory management and production planning. Wiley New York (1979)
32. Lin, C.T., Lee, C.S.G.: Neural-network-based fuzzy logic control and decision system. Computers, IEEE Transactions on **40**(12) (1991) 1320–1336
33. Yao, Y.Y.: A partition model of granular computing. Lecture Notes in Computer Science **3100** (2004) 232–253
34. Larose, D.T.: Discovering knowledge in data: an introduction to data mining. John Wiley & Sons (2014)
35. Zhao, Y., Yao, Y.Y., Luo, F.: Data analysis based on discernibility and indiscernibility. Information Sciences **177** (2007) 4959–4976
36. Liu, Q.H., Li, F., Min, F., Ye, M., Yang, G.W.: An efficient reduction algorithm based on new conditional information entropy. Control and Decision (in Chinese) **20**(8) (2005) 878–882
37. Liu, Q.H., Chen, L.T., Zhang, J.Z., Min, F.: Knowledge reduction in inconsistent decision tables. In: Advanced Data Mining and Applications. Springer (2006) 626–635
38. He, X., Min, F., Zhu, W.: Parametric rough sets with application to granular association rule mining. Mathematical Problems in Engineering **2013** (2013)
39. Pawlak, Z., Skowron, A.: Rough sets: some extensions. Information Sciences **177** (2007) 28–40
40. Boriah, S., Chandola, V., Kumar, V.: Similarity measures for categorical data: A comparative evaluation. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA. (2008) 243–254
41. Zhu, W., Wang, F.Y.: Reduction and axiomatization of covering generalized rough sets. Information Sciences **152** (2003) 217–230
42. Yang, T., Li, Q.: Reduction about approximation spaces of covering generalized rough sets. International journal of approximate reasoning **51** (2010) 335–345
43. Bianucci, D., Cattaneo, G., Ciucci, D.: Entropies and co-entropies of coverings with application to incomplete information systems. Fundamenta Informaticae **75**(1-4) (2007) 77–105
44. Bianucci, D., Cattaneo, G.: Information entropy and granulation co-entropy of partitions and coverings: A summary. Transactions on Rough Sets X **5656** (2009) 15–66
45. Min, F., Zhu, W.: Attribute reduction of data with error ranges and test costs. Information Sciences **211** (2012) 48–67
46. Wang, G.Y., Yu, H., Hu, F., et al.: Test-cost-sensitive attribute reduction in decision-theoretic rough sets. In: Multi-disciplinary Trends in Artificial Intelligence. Springer (2013) 143–152
47. Min, F., He, H.P., hua Qian, Y., Zhu, W.: Test-cost-sensitive attribute reduction. Information Sciences **181**(22) (2011) 4928–4942
48. Min, F., Liu, Q.H.: A hierarchical model for test-cost-sensitive decision systems. Information Sciences **179**(14) (2009) 2442–2452
49. Barakat, N.: Feature ranking utilizing support vector machines' svs. In: Innovative Computing Technology (INTECH), 2013 Third International Conference on, IEEE (2013) 401–406
50. Zhao, H., Zhu, W.: Optimal cost-sensitive granularization based on rough sets for variable costs. Knowledge-Based Systems **65** (2014) 72–82
51. Selakov, A., Cvijetinović, D., Milović, L., Mellon, S., Bekut, D.: Hybrid pso–svm method for short-term load forecasting during periods with significant temperature variations in city of burbank. Applied Soft Computing **16** (2014) 80–88
52. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Occam's razor. Information processing letters **24**(6) (1987) 377–380
53. Cimiano, P., Staab, S.: Learning by googling. ACM SIGKDD explorations newsletter **6**(2) (2004) 24–33
54. Lu, Q., Getoor, L.: Link-based classification. In: ICML. Volume 3. (2003) 496–503

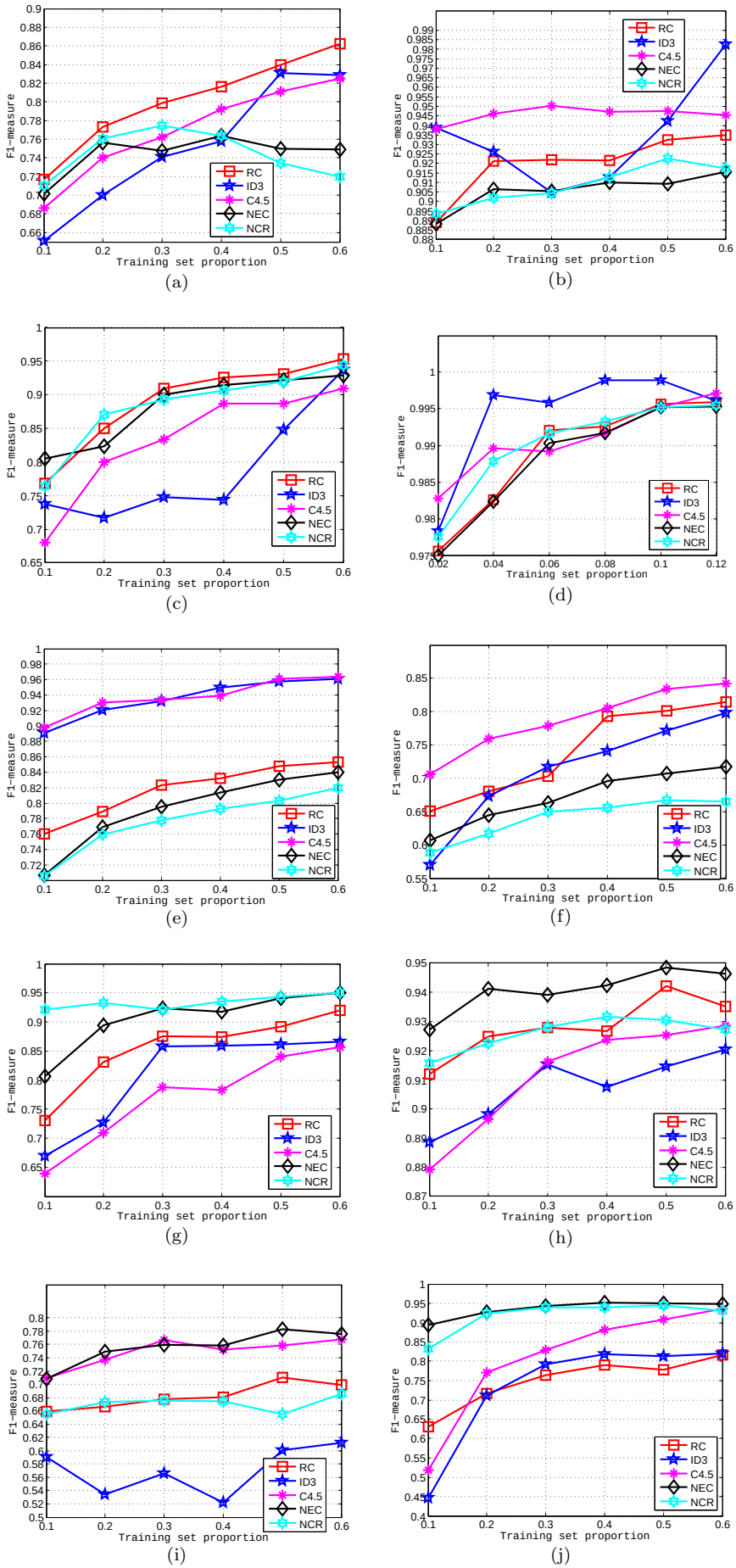


Fig. 6 Comparison of classification accuracies using the RC, ID3, C4.5, NEC, and NCR algorithms with (a) Tic-tac-toe, (b) Voting, (c) Zoo, (d) Mushroom, (e) Kr-vs-kp, (f) Splice, (g) Wine, (h) Wdbc, (i) Wpbc, (j) Iris.