Automated Case Selection from Databases Using Similarity-Based Rough Approximation

Liqiang Geng and Howard J. Hamilton

Department of Computer Science

University of Regina

Regina, Saskatchewan, Canada S4S 0A2

{gengl,hamilton}@cs.uregina.ca

Summary:

Knowledge acquisition for a case-based reasoning system from domain experts is a bottleneck in the system development process. It would be useful to derive representative cases automatically from larger, available databases rather than acquiring them from domain experts. Case selection is a branch of data mining that aims at choosing representative cases from large data sets for future case-based reasoning tasks. This chapter presents two algorithms using similarity-based rough set theory to derive cases automatically from available databases. The first algorithm, SRS1, requires the user to choose the similarity thresholds for the objects in a database, while the second algorithm, SRS2, can automatically select proper similarity thresholds. These algorithms can handle noise and inconsistent data in the database and select a reasonable number of the representative cases from the database. The algorithms were implemented and the experimental results showed that their classification accuracy was similar to that of well-known machine learning systems, such as rule induction systems, and neural network systems, and other state of art instance selection algorithms RT3 and ICF.

Keywords*: case selection, similarity, rough sets, classification.*

1.Introduction

Case-based reasoning (CBR) is a computerized method that attempts to study solutions that were used to solve problems in the past so as to solve current problems by analogy or association [10]. The CBR approach has some advantages over the rule-based approach in unstructured and difficult-to-understand problem domains, such as diagnosis, planning and classification. It can also deal with complex types of data that rule-based system cannot handle, such as images, sets and text. See for example [9, 4, 5, 17]. The success of these systems suggests that CBR-based systems are well suited to unstructured domains. Moreover, case bases are maintainable because new cases can be added and existing ones modified or deleted easily. However, knowledge acquisition for a CBR system is still a labor-intensive task. The classification accuracy of the developed CBR system is also dependent on the quality of the cases obtained from the domain experts.

Techniques for automatically deriving high quality cases from an available database are being studied by many CBR researchers. In this area, many issues have been and are being tackled, such as how to deal with noise and inconsistent data, how to reduce the number of the selected cases while maintaining classification accuracy, how to make cases more compact by eliminating irrelevant features, and how to increase the classification accuracy of the derived casebase.

If the cases have a simple structure with values for all of a set of attributes, such as a table in a relational database, then the traditional machine learning methods can be applied to obtain classification models. These approaches include *decision tree* learning systems, such as C4.5 [19]; rule learning systems, such as LEM2 [6]; and neural network models [20]. We now briefly review these approaches and then summarize other related research.

C4.5 is a recursive algorithm that builds a decision tree from the data available. The training set corresponds to the root node of the tree. If all training examples belong to the same class, the decision tree is a single leaf node. Otherwise, *information gain* is used as a heuristic to select an attribute, which partitions the training set into subsets according to the possible values of the attribute. These subsets form the child nodes for the root node. This partitioning process continues recursively until no node can be expanded further. The result is the decision tree.

LEM2 is based on rough sets theory, where a rough set is defined based on a lower approximation containing all objects certainly belonging to set and an upper approximation containing all objects possibly belonging to the set. Before the algorithm is applied, either the lower approximation or the upper approximation of the data set is calculated, depending on the type of rules desired. This process transforms inconsistent data into consistent data. Then for each decision class, the algorithm finds a local covering for the lower/upper approximation. An attribute-value pair is selected as part of the local covering based on heuristics, such as its

prespecified priority or its coverage. This selection process continues until the conjunction of the attribute-value pairs covers only positive examples. This conjunction forms the conditional part for a classification rule. After obtaining such a rule, the covered examples are removed from the training set. This process iterates until the training set is empty.

An *artificial neural network* (ANN) is an information-processing paradigm inspired by the way the interconnected, parallel structure of the mammalian brain processes information. The essence of the ANN paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements that are analogous to neurons. These elements (*neurons*) are tied together with weighted connections that are analogous to synapses. In the learning process, the examples are fed to the network and output is obtained. Based on the error between the actual output and the supposed output, the algorithm adjusts the connection weights (*synapses*). These connection weights store the knowledge necessary to solve specific problems.

Although these methods and algorithms have many advantages and researchers have developed many variations and descendents of these methods, they still can only be applied to well-structured domains.

Case based reasoning can deal with complex data types, whenever a similarity measure between data items can be defined. Case selection algorithms aim at selecting representative cases for future classification tasks. Case selection algorithms can be classified as either incremental or batch. An *incremental approach* decides whether to keep each new case in the casebase whenever a new case is encountered, while a *batch approach* creates an initial casebase from a data set or updates the casebase when its size grows to a certain threshold.

Aha et al. proposed a series of incremental instance-based learning algorithms (IBL), called IB1, IB2, IB3 and IB4 [1]. IB1 is similar to the k-nearest neighbor (k-NN) clustering algorithm, because it stores all the instances in the casebase and classifies unseen instances to the same class as the most similar instance in the casebase. IB2 reduces the size of the casebase by storing only the cases that cannot be correctly classified by the existing casebase. IB3 keeps track of the frequency with which stored cases, when chosen as one of the current object's most similar stored cases, matched the current object's decision value. If the matching frequency is lower than a given threshold, the case is discarded. IB4 incorporates weight learning and thus can tolerate irrelevant features better than IB3. The IBL algorithms have $O(mn^2)$ worst-case time complexity for m attributes and n instances. Due to their incremental nature, the casebase generated by the IBL algorithms is strongly influenced by the order in which the objects are stored. No heuristics are used to select instances from a global point of view.

To overcome this limitation, competence measures can be used to order the instances before the selection process begins. Smyth et al. propose *relative coverage* to measure the competence of instances [22]. This measure is not only

based on the coverage of an instance, but also based on how many other instances cover the covered instances. This ordering process transforms IBL from an incremental method into a batch method.

Similarly, Wilson et al. [24, 25] propose a series of batch case pruning algorithms named *RT*1, *RT*2, and *RT*3 (also called *Integrated Decremental Instance-Based Learning*). Instead of selecting good instances for inclusion, *RTx* selects bad ones for exclusion. *RT*1 first constructs lists of neighbors and lists of associates for each instance. Then it checks to see if the absence of each instance *i* will improve the classification accuracy of its *associates*, which are the instances that have *i* as one of their nearest neighbors. If it does, this instance is eliminated, and the lists of neighbors and associates are updated to delete this instance from their lists. *RT*2 does not update the associate lists; instead it keeps the deleted instances in the associate lists. Thus, the deleted instances continue to be considered during leave-one-out cross-validation. Furthermore, *RT2* ranks the instances according to their distances to the border in descending order. This sorting attempts to delete central points and keep border points. *RT3* attempts to eliminate noisy instances before the pruning process by deleting all instances that cannot be correctly classified by their nearest neighbors.

Brighton et al. [2] proposed a similar case pruning algorithm *ICF*. This algorithm uses the same method to identify and delete noisy instances as *RT*3. Then it calculates the reachable and coverage sets of each instance, which correspond to neighbors and associates in *RT*3. The difference between the reachable set and the neighbor set is that the neighbor set has fixed cardinality, while the reachable set has variable cardinality, which is determined by the *enemy* of the instance, i.e., the nearest neighbor of the instance that has a different decision value. The heuristic used here is that if the cardinality of the reachable set is greater than the cardinality of the coverage set, the instance is deleted. Although *RT*3 makes only one pass, *ICF* performs multiple passes until no further instances can be removed.

In addition to the case selection or case deletion strategies mentioned above, other approaches have also been proposed for maintaining large casebases. For example, Yang et al. [26] propose that a clustering algorithm be used to divide a large casebase into several smaller ones. When a new case arrives, information gain is used to identify the most similar casebase cluster, and then the standard case retrieval method is used to obtain the most similar cases from this cluster. Essentially, this method tends to maintain a layered casebase. It can improve the speed of case retrieval, but it may miss relevant cases and it does not reduce storage costs.

Hybrid case selection methods are also being studied. Li et al. show that k-NN and its variants perform poorly on nominal attributes [12]. To improve accuracy on data sets containing a mixture of interval-based and nominal attributes, they combine k-NN with the DeEPs algorithm. DeEPs classifies a new instance based on the support of subsets of the attribute-value pairs of the test instance. If the

total support for the subsets of the test instance which occur only in positive training instances is greater than the total support for those which occur only in negative instances, then the test instance is classified as positive, and vice versa. Using a similarity function, they define the neighborhood of a test instance. If the neighborhood contains training data, they apply k-NN to make a classification decision; otherwise, they use DeEPs [13].

Soft computing methods are also applied to case selection problems. Pal et al. use fuzzy set theory and rough set theory to induce representative cases from a database [16]. Their algorithm first translates the numeric data into fuzzy linguistic terms and then derives fuzzy rules from the database using a rough set approach. The fuzzy rules are then mapped to representative cases. Cao et al. use a similar method to derive representative cases [3]. They partition data into clusters, use a rough-fuzzy approach to obtain fuzzy adaptation rules for each cluster, and then select representative rules by applying these adaptation rules. As described, these algorithms can only deal with numeric data, which limits their applicability in case-based reasoning. Also these methods do not handle noisy or inconsistent data, and they require that many parameters and sub-algorithms be selected. The parameters and sub-algorithms include the number of the fuzzy terms for each attribute, thresholds for de-fuzzification, and fuzzy membership functions. Although these factors strongly influence the quality of the selected cases, they are difficult to choose. We believe that case generation should rely as much as possible on the available data rather than on information required from users.

In this chapter, two case selection algorithms are proposed. A similarity-based rough set method (SRS1) is suggested to select a reasonable number of representative cases from an available data set. It satisfies the following requirements: it selects the high quality center points, it has a small number of user-specified parameters and it explicitly handles noisy and inconsistent data. We also propose its variant (SRS2) that can determine the similarity threshold automatically. Section 2 discusses some basic concepts of similarity-based rough approximation, and section 3 introduces the concept of similarity measure. Sections 4 and 5 describe the proposed algorithms SRS1 and SRS2, respectively. Section 6 presents an illustrative example. In section 7, the results of testing the algorithms on well-known data sets are presented. Section 8 gives conclusions and directions for future work.

2. Similarity-Based Rough Sets

Rough sets [18] are a mathematical tool for dealing with vagueness and uncertainty in areas of artificial intelligence and cognitive sciences, such as data

mining, decision making, and pattern recognition [15, 8, 27]. The concept of rough sets is based on the assumption that every object of the universe can be represented by all information available about it. Objects represented by the same information are considered to be *indiscernible*. All the indiscernible objects form an elementary set, i.e., granule knowledge about the universe. If a given set of objects is a union of some elementary sets, it is referred to as a *crisp set*; otherwise it is a *rough set*. A rough set can be represented by a pair of crisp sets, called the *lower* and the *upper approximation*. The lower approximation contains all objects that surely belong to the set, and the upper approximation contains all objects that possibly belong to the set, with respect to the given knowledge.

Rough sets based on the indiscernibility relation can only deal with nominal attributes in the decision table [18]. Continuous attributes must be discretized into intervals and then each of these intervals must be translated into qualifiers before rough set theory can be applied. The discretization methods adopted greatly influence the quality of the classification results. A typical CBR based system usually involves more complex data types. Similarity measures are usually used for retrieving appropriate cases from the casebase. Therefore, we adopted the similarity relation-based rough set approach for selecting representative cases. Similarity relation-based rough set is an extension of the standard rough set approach, which replaces the indiscernibility relation with a similarity relation in the approximation process [21].

It is widely accepted that *similarity relation* should have *reflexive* properties, i.e., any object should be similar to itself. However, it is controversial whether similarity relation should be *symmetric*. Recently, more and more researchers tend to exclude the similarity property [21]. We argue that whether symmetry should be a property of the similarity relation depends on the context of application. Theoretically, we adopt the definition that similarity relation is reflexive, but not necessarily symmetric. However, practically, we might incorporate symmetric property for similarity relation, since this property can enhance computational efficiency, and any properties that hold for asymmetrical similarity relations also hold for symmetric ones.

Given a finite non-empty set U of objects, called the *universe*, a binary relation R defined on $U \times U$ is a similarity relation if and only if aRa, where $a \in U$. From this definition, we can represent the relation R as a similarity graph and define a similarity class for each object $x \in U$. The *similarity class* of x, denoted by $R(x)$, is the set of objects that are similar to x.

$$R(x) = \{y \in U \mid yRx\} \tag{1}$$

The *rough approximation* of a set $X \subseteq U$ is a pair of sets called lower and upper approximations of X, denoted by $R_*(X)$ and $R^*(X)$ respectively, where

$$R_*(X) = \{x \in X \mid R(x) \subseteq X\} \tag{2}$$

$$R^*(X) = \cup_{x \in X} R(x) \quad (3)$$

The lower approximation $R_*(X)$ of a set X is the set of objects whose similarity class belongs to X and the set consists of elements that can certainly be classified as elements of X, while the *upper approximation* $R^*(X)$ of X is the union of the similarity classes of objects in X and the elements in this set can possibly be classified as elements of X.

To make the approximation more robust, we make two extensions to the definition of the lower approximation.

First, to enable it to deal with noise in a database, we propose the following *extended lower approximation* of a similarity-based rough set,

$$R_{*ext}(X) = \{x \in X \mid card(R'(x)) / card(R(x)) \geq ct\}, \quad (4)$$

where $R'(x) = \{y \mid y \in R(x) \text{ and } d(y) = d(x)\}$ refers to the set of the objects that are similar to object x and have the same decision value as x. The *consistency threshold* $ct \in [0, 1]$ defines the degree to which noisy data are tolerated. It is a parameter that must be specified by the user for the SRS1 and SRS2 algorithms. $R_{*ext}(X)$ is the set of objects which are classified, with certainty of at least ct, as elements of concept X according to the similarity relation R. If ct equals 1, $R_{*ext}(X)$ is identical to the standard similarity based lower approximation $R_*(X)$.

Secondly, we think that in some cases, the quality of an object depends not only on the cardinality of the sets $R(x)$ and $R'(x)$, but also on the similarity measure between the object and its similar objects in sets $R(x)$ and $R'(x)$. Therefore, we propose the following extended *similarity-sensitive lower approximation*,

$$R_{*ss} = \{x \in X \mid \sum_{x_1 \in R'(x)} s(x, x_1) / \sum_{x_2 \in R(x)} s(x, x_2) \geq ct\}, \quad (5)$$

where $s(x, x_1)$ and $s(x, x_2) \in [0, 1]$ denote similarity degree between object x and objects x_1 and x_2 respectively. In the rest of the chapter, we use the first extension $R_{*ext}(X)$ and its relevant parameters to present our algorithms and examples. This can be easily replaced with the second extension $R_{*ss}(X)$ and its relevant parameters.

To conduct casebase generation, a source of data is represented as a *decision table*, i.e., a two-dimensional table where each row represents an object, and each column represents an attribute. The similarity relation is defined on the objects in the decision table where the cases are derived. Let $T = (U, A \cup \{d\})$ be a decision table where U is the universe, A is a set of condition attributes and d is a decision attribute [18]. In our current study, the condition attributes are restricted to be continuous, ordinal, or nominal and the decision attribute is nominal. Let V_a be a set of values of attribute a, where $a \in A$, $r(d)$ be the number of decision values, d_i be the *ith* decision value, and $Y_i = \{x \in U \mid d(x) = d_i\}$ be the set of objects that have

the *ith* decision value in the decision table. $POS(R,\{d\}) = \cup_{i=1}^{r(d)} R_{*ext}(Y_i)$ is called the *positive region* of the partition $\{Y_i \mid i = 1,...,r(d)\}$.

The coefficient $r(R, \{d\}) = card(POS(R, \{d\})) / card(U)$ is called the quality of approximation of the classification. It expresses the ratio of objects that can be correctly classified to all the objects in the decision table. The objects that cannot be classified are considered as inconsistent objects.

3. Similarity Measure

To obtain the similarity class for every object, a similarity measure should be defined for each attribute. This definition depends on the type of attributes under study. We classify the attributes into the following categories: interval-scaled attributes, ordinal attributes, nominal attributes, fuzzy attributes, set attributes, text attributes, and non-text attributes, such as images, multimedia data, etc. Many similarity measures have been proposed for these attributes, see for example [7, 23]. The most widely used categories are discussed as follows.

1. *Interval-scaled attributes* are continuous measurements on a roughly linear scale, such as length and height. For this kind of attribute, the similarity measure between object x and y $S_a(x, y)$ can be defined as:

$$S_a(x, y) = 1 - |a(x) - a(y)| / |a_{max} - a_{min}| \quad (6)$$

where x and y are the objects, a_{max}, a_{min} denote the minimum and maximum values of attribute a respectively, and $a(x)$, $a(y)$ denote the value of attribute a for object x and y respectively.

2. *Ordinal attributes* represent a set of states that are ordered in a meaningful sequence. Assuming that the values of attribute a are ordered in ascending order as $v_1, v_2, ...v_n$, and the objects x and y have the values v_i and v_j for attribute a, then S_a is defined as:

$$S_a(x, y) = 1 - |i - j| / (n - 1). \quad (7)$$

3. *Nominal attributes* have unordered discrete values. The following similarity measure is proposed:

$$S_a(x, y) = 1 - \sum_{k=1}^{r(d)} \frac{| P(d = k, a = a(x)) - P(d = k, a = a(y)) |}{r(d) P(d = k)}, \quad (8)$$

where $P(d = k,\ a = a(x))$ is the probability that the value of conditional attribute a is equal to the value of attribute a of object x and the decision is k, and $P(d = k,\ a = a(y))$ is likewise for object y.

4. *Fuzzy attributes* are a kind of nominal attributes that use fuzzy membership functions to define the semantics of the nominal terms. For example, the attribute *age* can take values of fuzzy terms *young*, *middle-aged* and *old*, each of which corresponds to a fuzzy membership function. If the attribute domain is discrete, the similarity measure can be defined as:

$$S_a(x, y) = 1 - \sum_{i=1}^{n} \frac{| u_A(t_i) - u_B(t_i) |}{n} \tag{9}$$

where A and B are the fuzzy terms of attribute a for objects x and y respectively and u_A and u_B are the corresponding fuzzy membership functions.

If the domain involves a continuous interval $[t_0,\ t_1]$, we define the similarity measure as:

$$S_a(x, y) = 1 - \int_{t_0}^{t_1} \frac{| u_A(t) - u_B(t) |}{t_1 - t_0} dt \tag{10}$$

5. *Set attributes* take sets for their values. For example, since an employee can have more than one child, the attribute *children* in an *employee* table could be a set attribute. One similarity measure for a set attribute is defined as:

$$S_a(x, y) = \frac{| v_x \cap v_y |}{| v_x \cup v_y |} \tag{11}$$

where v_x and v_y are attribute a's value for object x and y respectively.

6. *Text attributes* have text or documents for their values. A text value can be processed by information retrieval techniques to produce a set of keywords. Then we can use a similarity measure suited to set attributes to compute the similarity between two text values.

7. *Image attributes* have pictures in some prefined format as their values. They can be processed with the specific purpose distance functions.

After obtaining the similarity measure for each attribute, we aggregate them to define the global similarity measure on the set of objects by taking their product:

$$S(x, y) = \prod_{a \in A} S_a(a(x), a(y)), \tag{12}$$

or by taking the minimum similarity measure of the individual attribute:

$$S(x, y) = \mathrm{Min}_{a \in A}\ S_a(a(x), a(y)), \tag{13}$$

or by taking the algebraic average:

$$S(x, y) = \sum_{a \in A} S_a(a(x), a(y))/n, \quad (14)$$

where n is the number of the attributes,

or by taking the geometric average:

$$S(x, y) = \sqrt{\frac{\sum_{a \in A} S_a^2(x, y)}{n}} \quad (15)$$

or by taking their weighted sum:

$$S(x, y) = \sum_{a \in A} w_a S_a(a(x), a(y)). \quad (16)$$

where w_a is the weight for the attribute a, with $\sum_{a \in A} W_a = 1$

We say that object x is *similar* to object y if and only if the similarity measure between these two objects is greater than or equal to a similarity threshold *st*, i.e., xRy if and only if $S(x,y) \geq st$. The similarity threshold *st* determines the granularity of classification; a higher *st* value indicates a more refined classification of the data into clusters of representative cases. In algorithm SRS1, this parameter must be specified by the user, while in SRS2, this parameter is automatically determined by the algorithm.

4. SRS1 Algorithm

Based on the concept of similarity measure and rough set theory described above, we propose the following algorithm to find representative cases from a database. First, in addition to the similarity thresholds *st* and consistency threshold *ct* mentioned in section 2 and section 3, some variables used in the algorithm are defined as follows.

1. *SimilarNo*(i): the number of objects similar to the *ith* object O_i, including O_i itself.

2. *SimilarClassNo*(i): the number of objects similar to O_i that have the same decision value as O_i, including O_i itself.

3. *Consistency*(i)*:* the fraction of the number of similar objects that have the same decision value as O_i. *Consistency*(i) = *SimilarClassNo*(i) / *SimilarNo*(i). If *Consistency*(i) is less than *ct,* O_i can be regarded as inconsistent with the database.

4. $r(R, \{d\})$: the quality of classification, i.e., the fraction of the all objects that are consistent.

The steps of the algorithm is shown in Figure 1.

First, according to the similarity measures and the similarity threshold *st*, we construct the similarity relation graph of the objects, where each node represents an object in the data set. We then delete the inconsistent nodes according to the value of *Consistency* and threshold *ct*. Next we select the most representative node in terms of *Consistency* and *SimilarClassNo* and delete all of its similar nodes. The process iterates until the node set is empty. Meanwhile we determine the ratio $r(R, \{d\})$ of the objects that can be classified correctly to the whole set of objects. If the ratio is too low, which means too many objects cannot be classified, the threshold for the similarity measure is increased to refine the granularity of the clusters. In section 7, our experiments demonstrated that with a larger similarity threshold, more objects could be classified. But at the same time, the number of selected representative objects increases, which might give better classification accuracy.

Let m denote the number of the condition attributes and n denote the number of the objects in the decision table. The time complexity for creating a similarity relation graph is $O(mn^2)$, for computing *SimilarClassNo, SimilarNo* and *Consistency*, and selecting nodes is $O(n^2)$, and for deleting inconsistent nodes is $O(n)$. Overall, the complexity of the SRS1 is $O(mn^2)$. If $m << n$, the complexity approximates $O(n^2)$.

Algorithm SRS1(*st*, *ct*)
1. Based on *st*, compute the similarity measures between all pairs of objects and create a similarity matrix, which defines a similarity relation graph, with a node for each object and an arc between each pair of objects with nonzero similarity.
2. For each node *i* in the relation graph {
    Compute *SimilarNo*(*i*), *SimilarClassNo*(*i*), *and Consistency*(*i*).
}
3. Compute $r(R, \{d\})$.
4. Delete nodes that are considered inconsistent, i.e., those which satisfy *Consistency*(*i*) < *ct*
5. Select isolated nodes, i.e., nodes whose *SimilarNo* is 1, and place them in the casebase.
6. While the node set is not empty {
    Select a node with the maximum value for *Consistency*. If there is a tie, select one with the maximum value for *SimilarClassNo*. If there is a tie again, randomly select one node. Insert it into the casebase.
    Delete the selected node and all nodes adjacent to it.
}

Fig. 1. Algorithm SRS1

5. SRS2: Automatic Determination of Similarity Threshold

It is often difficult for users to determine an appropriate value for the similarity threshold *st*. Therefore this threshold is usually obtained through trial and error. We propose that a more precise and robust definition of similarity can be inferred directly from the data sets. Krawiec et al. [11] proposed a method to derive the discretization intervals for each conditional attribute in the decision rules. Their method is strongly local in that it computes the intervals for each conditional attribute independently from the rest of the conditional attributes. In our method, the entire set of conditional attributes is simultaneously considered. We take into account the similarity between objects. The algorithm is presented in Figure 2.

This algorithm assumes that the similarity threshold for each object is unique and that these thresholds can be determined on the basis of the consistency threshold *ct* provided by the users. In this algorithm, we first compute the similarity matrix for the given set of objects. Then for each object we sort the other objects that are similar to it in descending order according to the similarity measure. The similarity threshold for O_i should satisfy the consistency condition $SimilarClassNo(i) / SimilarNo(i) \geq ct$. Therefore, we select the similar objects in the ordered list one by one as long as the ratio is greater than or equal to the given threshold *ct*. Then, for the selected objects, we sort them in ascending order according to the similarity to O_i. In this order, we delete the objects one by one until the current object in the list has the same decision value as O_i, and at the same time, the consistency condition holds. In this way, we obtain the set of similar objects to O_i. Finally we construct the similarity relation graph, which in this case is a directed graph. Then we obtain the representative cases with the same method as in SRS1 described in section 4.

Let *m* denotes the number of the condition attributes and *n* denotes the number of the objects in the decision table. The time complexity for creating a similarity matrix is $O(mn^2)$, that for sorting nodes according to similarity measure is $O(n^2 logn)$, that for computing *SimilarClassNo, SimilarNo* and *Consistency*, and for selecting nodes is $O(n^2)$, and that for deleting the inconsistent nodes is $O(n)$. Overall, the complexity of the SRS1 algorithm is $O(mn^2 logn)$. If $m << n$, the complexity approximates $O(n^2 logn)$.

Algorithm SRS2(*ct*)

1. Compute the similarity measure between all pairs of objects and create a similarity matrix, which defines a similarity graph.
2. For each object O_i, do {
    Create an ordered list by sorting all objects in descending order according to their similarity recorded in the similarity matrix to object O_i
    SimilarNo(*i*) = 0;
    SimilarClassNo(*i*) = 0;
    SimilarSet(*i*) = {};
    For each object O_j in the ordered list, do the following {
        SimilarNo(*i*) = *SimilarNo*(*i*) + 1;
        SimilarSet(*i*) = *SimilarSet*(*i*) $\cup$ O_j;
        If the decision value of object O_j equals the decision value of O_i
            SimilarClassNo(*i*) = *SimilarClassNo*(*i*) + 1;
        Consistency(*i*) = *SimilarClassNo* / *SimilarNo;*
        If *Consistency*(*i*) < *ct*
            Break;
    }
    For each object O_j in *SimilarSet(i)*, in ascending order in terms of the similarity measure, do the following {
        If the decision value of object O_j is not equal to the decision value of O_i {
            SimilarSet(*i*) = *SimilarSet*(*i*) - O_j;
            SimilarNo(*i*) = *SimilarNo*(*i*) - 1;
            Consistency(*i*) = *SimilarClassNo*(*i*) / *SimilarNo*(*i*);
        }
        Else if *Consistency*(*i*) < *ct* {
            SimilarSet(*i*) = *SimilarSet*(*i*) - O_j;
            SimilarNo(*i*) = *SimilarNo*(*i*) - 1;
            SimilarClassNo(*i*) = *SimilarClassNo*(*i*) - 1;
            Consistency(*i*) = *SimilarClassNo*(*i*) / *SimilarNo*(*i*);
        }
        Else
            Break;
    }
}
3. While the set of the objects is not empty {
    Select isolated nodes to the casebase, i.e., nodes whose *SimilarNo* is 1.
    Select a node with the maximum value for *Consistency*. If there is a tie, select one with the maximum value for *SimilarClassNo*. If there is a tie again, randomly select one node. Insert it into the casebase.
    Delete the selected node and all nodes adjacent to it.
}

Fig. 2. Algorithm SRS2

6. An Illustrative Example

In this section, we present an example to illustrate the two algorithms. Table 1 has been adapted from [21] and consists of information about eleven companies. The condition attributes are *asset* (*a*), *profit* (*p*), and *type of product* (*t*), and the decision attribute is *credit* (*c*). *Asset* and *profit* are interval scaled values, *product* is a nominal attribute, and *credit* is a nominal attribute.

Here we define the similarity measures for the condition attributes:

$S_a(V_{ai}, V_{aj}) = 1 - |V_{ai} - V_{aj}| / |V_{amax} - V_{amin}|$

$S_p(V_{pi}, V_{pj}) = 1 - |V_{pi} - V_{pj}| / |V_{pmax} - V_{pmin}|$

$S_t(V_{ti}, V_{tj}) = 1$, if $V_{ti} = V_{tj}$; $S_t(V_{ti}, V_{tj}) = 0$, otherwise

$S(O_i, O_j) = S_a(V_{ai}, V_{aj}) * S_p(V_{pi}, V_{pj}) * S_t(V_{ti}, V_{tj})$.

where O_i and O_j denote objects i and j respectively, V_{ai} and V_{aj} denote the values for O_i and O_j for attribute a respectively, V_{amax} and V_{amin} denote the maximum and minimum values for attribute a in the decision table. Similar notation is defined for attributes p and t. S_a, S_p, and S_t denote the similarity measure between objects in terms of the attributes a, p, and t, respectively. S denotes the overall similarity measure between objects.

Table 1. Decision table

Company	Asset	Profit	Type of Product	Credit
1	105	67	Computer software	Bad
2	54	75	Automobile	Good
3	80	93	Automobile	Bad
4	64	80	Automobile	Good
5	92	92	Computer hardware	Good
6	96	102	Computer hardware	Good
7	111	65	Computer software	Bad
8	58	70	Automobile	Good
9	74	77	Automobile	Bad
10	105	105	Computer hardware	Good
11	85	82	Automobile	Bad

By calculating the similarity measure for all pairs of objects, we obtain the similarity matrix shown in Table 2.

6.1 Example of Algorithm SRS1

With the similarity matrix shown in Table 2, we can apply algorithm SRS1 to the example in Table 1. If the similarity threshold *st* is set to 0.65, then we obtain the relation graph shown in Figure 3. The objects in the left area are the companies with *bad credit*, while those in the right area are the companies with *good credit*.

Table 2. Similarity matrix

Object	1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	0	0	0	0.85	0	0	0	0
2		1	0.3	0.72	0	0	0	0.81	0.62	0	0.38
3			1	0.38	0	0	0	0.26	0.54	0	0.66
4				1	0	0	0	0.67	0.76	0	0.6
5					1	0.7	0	0	0	0.52	0
6						1	0	0	0	0.78	0
7							1	0	0	0	0
8								1	0.6	0	0.37
9									1	0	0.71
10										1	0
11											1

If we set the consistency threshold *ct* to 0.7, the *SimilarNo, SimilarClassNo* and *Consistency* values for each object can be computed as shown in Table 3. Take object 9 for example. The similar objects for object 9 are objects 4, 9, and 11, in other words, similarity measures between object 9 and these three objects are greater than *st*, therefore *SimilarNo*(9) is calculated as 3. Among these similar objects, objects 9 and 11 have the same decision value, *bad credit*, as object 9. Object 4 has a different decision value. Therefore, *SimilarClassNo* is calculated as 2. Then *Consistency*(9) is calculated as 2/3 (0.67). Similarly, we obtain *Consistency*(4) = 0.75, and for the other nodes, *Consistency* = 1. For node 9, we can say that if a node is similar to it, it can be classified into the *bad credit* group

only with a probability of 0.67, which is less than the threshold *ct*, 0.7. Hence we conclude that node 9 is an inconsistent object and should be eliminated from the graph. We then examine the nodes with the maximum value for *Consistency*. In this case nine nodes tie, and therefore we select a node with the maximum value for *SimilarClassNo*, i. e., one node from among the nodes 2, 6, 8, and 11. Since there are more than one candidate nodes, we randomly select one of them, say node 2, into the casebase. We then delete the nodes connected to node 2, i.e., node 8 and node 4. Next we select node 6 and delete node 5 and node 10. In the third iteration, we select node 11 and delete nodes 3 and 9. Finally we select node 1 and delete node 7. In all, nodes 1, 2, 6, and 11 are selected and placed in the casebase.

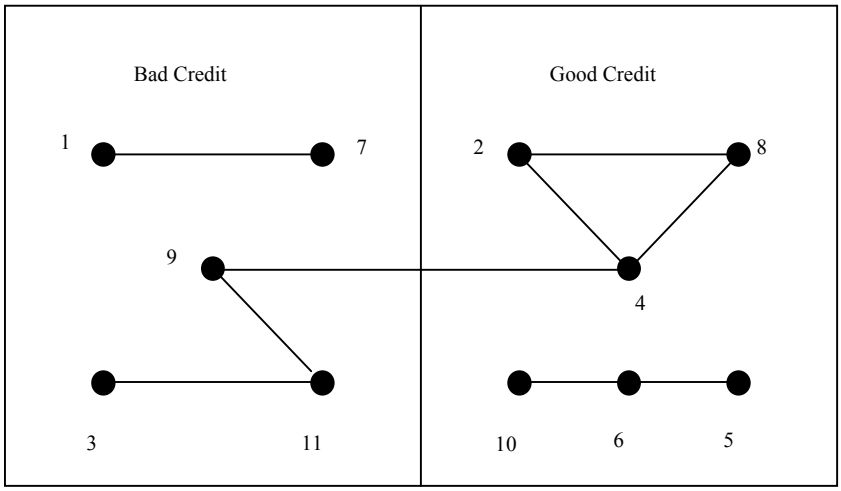

Fig. 3. Similarity relation graph.

6.2 Example of Algorithm SRS2

The consistency threshold is defined as *ct* = 0.7. For each object, we sort all the objects in descending order according to the similarity measure, and identify the set of similar objects according to *ct.* Table 4 presents the set of similar object and other parameters for each object. Using the same selection strategy described for step 4 of the SRS1 algorithm, objects 2, 3, 5, and 1 are selected as the representative cases.

Table 3. Sets of similar objects with associated *SimilarNo*, *SimilarClassNo*, and *Consistency* values for example of SRS1.

Object	*SimilarNo*	*SimilarClassNo*	*Consistency*	Above *ct*
1	2	2	1	yes
2	3	3	1	Yes
3	2	2	1	Yes
4	4	3	0.75	Yes
5	2	2	1	Yes
6	3	3	1	Yes
7	2	2	1	Yes
8	3	3	1	Yes
9	3	2	0.67	No
10	2	2	1	Yes
11	3	3	1	Yes

Table 4. Sets of similar objects with associated *SimilarNo*, *SimilarClassNo* and *Consistency* values for example of SRS2.

Object	Set of Similar Objects	*SimilarClassNo*	*SimilarNo*	*Consistency*
1	1, 7	2	2	1
2	2, 8, 4	3	3	1
3	3, 11, 9	3	3	1
4	4	1	1	1
5	5, 6, 10	3	3	1
6	6, 10, 5	3	3	1
7	7, 1	2	2	1
8	8, 2, 4	3	3	1
9	9	1	1	1
10	10, 6, 5	3	3	1
11	11, 9, 3	3	3	1

7. Experimental Results

SRS1 and SRS2 algorithms were implemented in Java and experiments were conducted on a PC with 128Mb memory and 733MHz CPU speed. Each run of the two algorithms on all testing data sets takes less than one minute.

In the experiments, we first applied the algorithms to three well-known data sets: Iris (Fisher's Iris Plant Database), Glass (Glass Identification Database) and Pima (Pima Indians Diabetes Database) to study the interdependency between the classification accuracy, storage and the parameters *ct* and *st* . These data sets are available from the University of California Repository of Machine Learning Databases [14] and are widely used by the machine learning community for empirical analysis of algorithms. The Iris data set has four condition attributes, three decision values and 150 objects; the Glass data set has nine condition attributes, six decision values and 214 objects; and the Pima data set has eight condition attributes, two decision values and 768 objects.

We adopted 10-fold cross validation method to evaluate the accuracy of the learned classification systems. The classification accuracy of the derived casebase is obtained in the following way. We divided the data set into 10 equal subsets. Then selected one subset as the testing set, and the union of the other nine sets as training set. We repeated this process a total of ten times. After generating the casebase from the training set, the system then classifies the testing set according to the similarity-based casebased reasoning algorithm.

The experimental results of SRS1 on the Iris data set are shown in Table 5. The first number in each grid of the table represents the number of the selected cases, the second one represents the average classification accuracy on the test data set, and the third one represents the standard deviation of the classification accuracy. Experimental runs were conducted using SRS1 with different similarity and consistency thresholds. The results for the run(s) with the highest classification accuracy are shown in boldface. From the experiments, it was observed that when the similarity threshold *st* increases, the granularity of the classification becomes finer, the number of the selected cases increases, and the classification accuracy usually, but not always, increases. When the consistency threshold *ct* increases, the system tolerates less noisy data and identifies more objects as noise. Therefore, the number of the selected cases decreases. For the Iris data set, we did not observe any interdependency between *ct* and the classification accuracy.

Table 6 presents the experimental results of SRS2 on the three data sets. As in Table 5, the first number in each grid represents the number of the selected cases, and the second and third numbers represent the classification accuracy and standard deviation of the derived casebase, respectively. However, for SRS2, only one parameter, the consistency threshold *ct,* requires definition. When *ct* increases, it is observed that the number of the selected cases increases. This result is in contrast to the experiments with SRS1, where the number of the selected cases

decreases when *ct* increases. This is because in SRS1, the similarity threshold and consistency threshold are given by the user and they have no inter-dependency, while in SRS2, the similarity threshold is determined directly by the consistency threshold. When the consistency threshold increases, the tolerance for noisy data decreases. Therefore, in the process of selection, the granularity of the clusters of cases becomes finer, and more objects are selected. In this experiment, we also observe that the optimal *ct* value depends on the data set itself.

Table 5. Experimental results of running SRS1 on the Iris data set.

ct\st	0.5	0.6	0.7	0.8	0.9
0.5	8, 92.7, 7.6	13, 94.7, 5.0	**20, 98.0, 4.3**	40, 95.3, 4.3	92, 93.3, 4.2
0.6	8, 92.0, 7.8	13, 96.0, 4.4	**20, 98.0, 4.3**	40, 96.0, 3.3	92, 93.3, 4.2
0.7	7, 88.7, 7.9	13, 96.0, 4.4	**20, 98.0, 4.3**	40, 96.0, 3.3	92, 93.3, 4.2
0.8	6, 89.3, 6.8	12, 96.7, 3.3	**20, 98.0, 4.3**	40, 96.0, 3.3	92, 93.3, 4.2
0.9	6, 90.0, 5.4	10, 95.3, 6.0	18, 96.7, 4.5	38, 96.0, 3.3	92, 93.3, 4.2
1.0	6, 90.7, 5.3	10, 94.0, 6.3	18, 96.7, 4.5	38, 96.0, 3.3	92, 93.3, 4.2

Table 6. Experimental results of SRS2 on three data sets.

ct\data set	Iris	Glass	Pima
0.5	5, 78.7, 9.3	40, 47.6, 8.5	3, 34.2, 6.0
0.6	5, 79.3, 9.2	52, 47.7, 11.1	4, 39.7, 9.6
0.7	4, 75.3, 9.5	56, 53.7, 5.1	20, 59.0, 10.5
0.8	6, 73.3, 8.4	61, 62.1, 7.2	44, 65.5, 9.1
0.9	5, 88.0, 11.1	**76, 63.5, 7.5**	70, 68.5, 7.3
1.0	**10, 96.7, 3.3**	75, 63.2, 6.3	**127, 73.0, 6.6**

The performance of SRS1 and SRS2 was compared with other classification systems. Table 7 compares the classification accuracy obtained using the SRS1 and SRS2 algorithms with those of four well known data mining systems: the tree induction algorithm C4.5 [19], layered Artificial Neural Network (ANN) [20], Instance Based Learning 3 (IB3) [1] and rule induction algorithm LEM2 [6]. The data in the first four rows were obtained from [11]. The data of the last two rows are the best average accuracy for SRS1 and SRS2 obtained using the testing methods described above in this section.

Table 7. Comparison of the accuracy with other classification algorithms.

Algorithm	Iris	Glass	Pima
C4.5	95.5	67.9	70.8
ANN	95.3	65.0	**76.4**
IB3	96.7	65.4	68.2
LEM2	94.0	66.8	62.0
SRS1	**98.0**	**69.7**	74.2
SRS2	96.7	63.5	73.0

From Table 7, it can be seen that the system classification accuracy for both SRS1 and SRS2 algorithms approximate that of other well-known classification systems, provided the parameters are properly selected.

Finally we compare the performance of SRS1 and SRS2 with other state of art case selection algorithms, RT3 and ICF. Table 8 summarizes the classification accuracy (Acc) obtained and storage required using RT3, ICF, SRS1, and SRS2, on nine data sets. The results for RT3 and ICF were obtained from [2].

Table 8. Comparison of accuracy and storage with *RT*3 and *ICF*

Dataset	RT3		ICF		SRS1		SRS2	
	Acc	Storage	Acc	Storage	Acc	Storage	Acc	Storage
Balance-scale	83.4	18.2	81.5	14.7	**90.0**	**5.2**	79.1	**5.2**
Breast cancer-w	95.3	3.1	95.1	4.3	**96.4**	3.3	95.1	**2.2**
Ecoli	**82.8**	15.8	81.3	14.0	76.5	**6.6**	73.6	31.5
Glass	69.1	**23.3**	69.6	31.4	**69.7**	47.7	63.5	39.4
Iris	93.6	16.0	92.6	42.0	**98.0**	14.8	96.7	**7.4**
Pima	71.0	22.4	69.2	**17.2**	**74.2**	18.7	73.0	18.4
Voting	93.8	7.4	91.2	8.9	**94.0**	**1.3**	92.2	7.9
Wine	86.4	15.4	83.8	12.0	**95.5**	8.8	94.3	**8.1**
Zoo	87.1	26.1	92.4	52.8	**96.1**	26.4	89.3	**11.0**

8. Conclusion

In this chapter, we proposed two similarity-based rough set algorithms called SRS1 and SRS2 for selecting representative cases from data sets. These algorithms are elegant in that they need only one or two parameters to be specified before the selection process. SRS1 requires the user to input the consistency threshold and the similarity threshold. SRS2 requires only the consistency threshold from the user, because it can automatically choose an appropriate similarity threshold for each object. Therefore, it requires fewer runs and saves the user's time. Furthermore, SRS2 generates fewer cases than SRS1. Both SRS1 and SRS2 can deal with noise and inconsistent data in the database. The selected cases are representative in that they can cover all of the data in the data set in terms of the similarity relation, and at the same time, these cases are not similar to each other. The experimental results indicate that the classification accuracy for both systems is comparable to classical machine learning systems.

Another advantage of these algorithms is that they are highly scalable. We can divide a large data set into smaller ones, and select the cases from those subsets by applying these algorithms. Then, collecting all the selected cases together, we apply the algorithms again to select cases for the second round. Take SRS1 for example. Suppose there are m attributes and n objects in the data set. We divide it evenly into p subsets. Each subset has $q = [n/p]$ objects. So, applying SRS1 to each subset, we have complexity $O(mq^2)$. For all the subsets, complexity is $O(mnq)$. Then suppose we selected n' cases in total, then working on it, the complexity for the second round is $O(mn'^2)$. In total, the complexity is $O(mnq+mn'^2)$. When q and n' are far less than n, the complexity is improved for $O(mn^2)$.

In the experiments reported here, we only used accuracy first case selection strategy. In the future we will adopt other heuristics, such as coverage first strategy or dynamic selection strategy, and try more similarity measures. We will apply these algorithms to data sets containing complex data types. We will also incorporate attribute reduction and weight assignment algorithms to make the generated cases more compact.

References

1. Aha, D.W., Kibler, D., and Albert, M.K., Instance based learning algorithms. *Machine Learning*, 6, 37-66, 1991.
2. Brighton, H. and Mellish, C., Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, 6, 153-172, 2002.
3. Cao, G., Shiu, S., and Wang, X., A fuzzy-rough approach for case base maintenance. *Proceedings of the Fourth International Conference on Case-Based Reasoning*, 118-130, 2001.

4. Chan, C., Chen, L., and Geng, L., Knowledge engineering for an intelligent case-based system for help desk operations. *Expert Systems with Applications*, 18, 125-132, 2000.
5. Ganesan, K., Khoshgoftaar, T.M., and Allen, E.B., Case-based software quality prediction. *International Journal of Software Engineering and Knowledge Engineering*, 10 (2), 139-152, 2000.
6. Grzymala-Busse, J.W., LERS - a system for learning from examples based on rough sets. Slowinski, R. (ed.), *Intelligent Decision Support*, Kluwer Academic, 3-18, 1992.
7. Han, J. and Kamber, K. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.
8. Hu, X. and Cercone, N., Rough sets similarity-based learning from databases. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*. 162-167, 1995.
9. Jurisica, I. and Glasgow, J., Extending case-based reasoning by discovering and using image features in IVF. *Proceedings of the 16th ACM SAC2001 Symposium on Applied Computing*, 52-59, 2000.
10. Ketler, K. Case based reasoning: an introduction. *Expert Systems with Applications*, 6, 3-8, 1993.
11. Krawiec, K., Slowinski, R., and Vanderpooten, D., Learning decision rules from similarity based rough approximations. Polkowski, L, Skowron, A. (eds.), *Rough Sets in Knowledge Discovery*, vol. 2, Heidelberg; New York: Physica-Verlag, 37-54, 1998.
12. Li, J., Dong, G., and Ramamohanarao, K., Instance-based classification by emerging patterns. *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases*. 191-200, 2000.
13. Li, J., Ramamohanarao, K., and Dong, G., Combining the strength of pattern frequency and distance for classification. *Proceedings of the Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 455-466, 2001.
14. Merz, C.J. and Murphy, P.M., UCI Repository of machine learning databases. University of California, Irvine, Department of Information and Computer Science, 1996.
15. Mrozek, A. and Skabek, K. Rough sets in economic applications. Polkowski, L., Skowron, A. (eds.), *Rough Sets in Knowledge Discovery*, vol. 2, Heidelberg; New York: Physica-Verlag 238-271, 1998.
16. Pal, S. K. and Mitra, P., Case generation: a rough-fuzzy approach. *Proceedings of Workshop Program at the 4th International Conference on Case-Based Reasoning*, 236-242, 2001.
17. Park, H., Oh, J., Jeong, D, and Park, K., Automated sleep stage scoring using hybrid rule- and case-based reasoning. *Computers and Biomedical Research*, 33(5), 330-349, 2000.
18. Pawalk, Z., *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer Academic, Dordrecht, 1991.
19. Quinlan, J.R., *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo CA, 1988.
20. Rumelhart, D.E., Hinton, G.E., and Williams, R.J., Learning internal representations by error propagation. Rumelhart, D.E, McClelland, J.L. and the PDP Research Group (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, Cambridge, MA, 318-362, 1986.
21. Slowinski, R. and Vanderpoonten, D., Similarity relation as a basis for rough approximations. *Advances in Machine Intelligence and Soft Computing,* 4, 17-33, 1997.

22. Smyth, B. and McKenna, E., Building compact competent case-bases. *Proceedings of the Third International Conference on Case-based Reasoning*, 329-342, 1999.
23. Wang, W., New similarity measures on fuzzy sets and on elements. *Fuzzy Sets and Systems*, 85, 305-309, 1997.
24. Wilson, D.R. and Martines, T.R., An integrated instance-based learning algorithm. *Computational Intelligence*, 16(1), 1-28, 2000.
25. Wilson, D.R. and Martines, T.R., Instance pruning techniques. *Proceedings of the Fourteenth International Conference on Machine Learning* 404-411, 1997.
26. Yang, Q. and Wu J., Keep it simple: a case-base maintenance policy based on clustering and information theory. *Proceedings of the Thirteenth Canadian AI Conference*, 102-114, 2000.
27. Ziarko, W., Probabilistic decision tables in the variable precision rough set model. *Computational Intelligence*, 17(3), 593-603, 2001.