Vietnam J Comput Sci (2014) 1:47–55
DOI 10.1007/s40595-013-0004-3

REGULAR PAPER

Feature selection and replacement by clustering attributes

Tzung-Pei Hong · Yan-Liang Liou · Shyue-Liang Wang · Bay Vo

Received: 2 October 2013 / Accepted: 3 October 2013 / Published online: 9 November 2013

Abstract Feature selection is to find useful and relevant features from an original feature space to effectively represent and index a given dataset. It is very important for classification and clustering problems, which may be quite difficult to solve when the amount of attributes in a given training data is very large. They usually need a very time-consuming search to get the features desired. In this paper, we will try to select features based on attribute clustering. A distance measure for a pair of attributes based on the relative dependency is proposed. An attribute clustering algorithm, called Most Neighbors First, is also presented to cluster the attributes into a fixed number of groups. The representative attributes found in the clusters can be used for classification such that the whole feature space can be greatly reduced. Besides, if the values of some representative attributes cannot be obtained from current environments for inference, some other possible attributes in the same clusters can be used to achieve approximate inference results.

Keywords Attribute clustering · Feature selection · Representative attribute · Relative dependency

This is an extended version of the paper presented in The Sixth International Conference on Machine Learning and Cybernetics.

T.-P. Hong
Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan

Y.-L. Liou
Department of Electrical Engineering, National University of Kaohsiung, Kaohsiung, Taiwan
e-mail: egghead221@gmail.com

S.-L. Wang
Department of Information Management, National University of Kaohsiung, Kaohsiung, Taiwan
e-mail: slwang@nuk.edu.tw

T.-P. Hong (✉)
Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan
e-mail: tphong@nuk.edu.tw

B. Vo
Ton Duc Thang University, Ho Chi Minh, Vietnam
e-mail: vdbay@it.tdt.edu.vn

1 Introduction

Although a wide variety of expert systems has been built, knowledge acquisition remains a development bottleneck [2,21]. Building a large-scale expert system involves creating and extending a large knowledge base over the course of many months or years. Shortening the development time is thus the most important factor for the success of an expert system. In the past, machine-learning techniques were successfully developed to ease the knowledge-acquisition bottleneck. Among the proposed approaches, deriving rules from training examples is the most common [9,14,15]. Given a set of examples, a learning program tries to induce rules that describe each class.

In some application domains, the amount of attributes (or features) of given training data is very large (e.g. decades to hundreds). In this case, much computational time is needed to derive classification rules from the data. Besides, derived rules may contain too many features and more rules than actually desired may be obtained due to over-specialization. In fact, not all the attributes are indispensable. Some redundant, similar or dependent attributes may exist in the given training data. This phenomenon mainly results from attribute dependency. Redundant and similar attributes can be thought of as two special cases of dependent attributes. If there exists

some dependency relationship between attributes, the dimension of the training data may thus be reduced.

The concept of reduced attributes has been used in many places. For example, in the rough set theory [16–19], the reduced set of attributes is also called a "reduct". Many possible reducts may exist at the same time. Even only a reduced set of attributes is used for classification, the indiscernibility relations still preserve among the attributes [10]. A minimal reduct, just as its literal meaning shows, is a reduct which cannot be reduced any more. It may not be unique as well. The classification work for a high-dimensional dataset can be done faster if a minimal reduct instead of the original entire set of attributes is used. Finding a minimal reduct is an NP-hard problem [20,22]. Besides, there may be no minimal reduct due to noise in training examples.

Some researches about finding approximate reducts were thus proposed. An approximate reduct is a minimal reduct with acceptable tolerance. It can usually be found in much shorter time relative to an exact minimal reduct. Besides, it usually consists of less attributes than an exact one. It is thus a good trade-off among accuracy, practicability and execution time. Many approaches for finding approximate reducts were proposed [3,7,26,27]. For example, Wróblewski [25] used the genetic algorithm to find approximate minimal reducts. Sun and Xiong [23] proposed an approach compatible with incomplete information systems. Al-Radaideh et al. [1] used the discernibility matrix and a weighting strategy to find the minimal reduct in a greedy strategy. Gao et al. [4] proposed a feature ranking strategy (similar to attribute weighting) with a sampling process included. Recently, approaches based on soft sets for attribute selection has also been proposed to reduce the execution time [13,20].

All the approaches mentioned above focus on the issue of finding a minimal reduct as soon as possible. However, if there are training examples with missing or unknown values, the approach may not correctly work. Besides, if only the chosen reduct is used in a learning process, the rules cannot contain other attributes and are hard to use if some attribute values in the reduct cannot be obtained in current environments.

In this paper, we solve the above problems from another viewpoint–attribute clustering. Note that we are doing clustering for attributes rather than for objects. Like the conventional clustering approaches for objects, the attributes within the same cluster are expected to possess high similarity, but within different clusters possess low similarity. Here, the dependency degrees between attributes are used to represent the similarities. Since the attributes are grouped into several clusters according to their similarity degrees, an attribute selected from a cluster can thus represent the attributes within the same cluster. An approximate reduct could then be formed from the chosen attributes gathered together. Note that the obtained result in this way is usually an approximate reduct. The proposed approach has the following three advantages.

1. Guessing a missing value of an attribute from the other attributes within the same cluster should be more accurate and faster than that from all attributes.
2. If an object has missing values, its class can also be decided by the other attributes within the same cluster.
3. The proposed approach is flexible for representing rules since each attribute in a rule can be displaced with other attributes in the same cluster.

Besides, the proposed algorithm for clustering attributes is also implemented to verify its effects. Experimental results show that the average similarity of each cluster is related to the cluster number. As the cluster number increases, the average similarity of a cluster will also increase.

The remainder of this paper is organized as follows: Some related concepts including reduct, relative dependency and clustering are reviewed in Sect. 2. The proposed dissimilarity between a pair of attributes is explained in Sect. 3. An attribute clustering algorithm is proposed in Sect. 4. An example is given in Sect. 5 to illustrate the proposed algorithm. The experimental results and some discussions are described in Sect. 6. Conclusions and future work are finally stated in Sect. 7.

2 Related work

In this section, some important concepts related to this paper are briefly reviewed. The concept of reducts is first introduced, followed by the concept of relative dependency. Next, two famous clustering approaches, k-means and k-medoids, are described and compared. The reasons for why they are not suitable for clustering attributes are also described. An attribute clustering approach is thus proposed due to these problems and limitations.

2.1 Reducts

Let $I = (U, A)$ be an information system, where $U = \{x_1, x_2, \ldots, x_N\}$ is a finite non-empty set of objects and A is a finite non-empty set of attributes called condition attributes [10]. A decision system is an information system of the form $I = (U, A \cup \{d\})$, where d is a special attribute called decision attribute and $d \notin A$ [10]. For any object $x_i \in U$, its value for a condition attribute $a \in A$, is denoted by $f_a(x_i)$. The indiscernibility relation for a subset of attributes B is defined as:

$$\mathrm{IND}(B) = \{(x, y) \in U \times U \mid \forall a \in B,\ f_a(x) = f_a(y)\},$$

where B is any subset of the condition attribute set A (i.e. $B \subseteq A$) [11]. If the indiscernibility relations from both A and

B are the same [i.e. IND(B) = IND(A)], then B is called a reduct of A. That means the attributes used in the information system can be reduced to B, with the original indiscernibility information still kept. Furthermore, if an attribute subset B satisfies the following condition, then B is called a minimal reduct of A:

$$\text{IND}(B) = \text{IND}(A) \quad \text{and} \quad \forall B' \subseteq B \ \ \text{IND}(B') \neq \text{IND}(A).$$

Take the simple information system in Table 1 as an example. In Table 1, the attribute set A consists of three attributes {Age, Income, Children} and the object set U consists of five objects $\{x_1, x_2, x_3, x_4, x_5\}$. Since IND({Age, Children}) = IND(A) = $\{(x_1, x_1), (x_2, x_2), (x_3, x_3), (x_4, x_4), (x_5, x_5)\}$, the attribute subset {Age, Children} is a reduct of the information system. Besides, since neither IND({Age}) nor IND({Children}) equals IND(A), the attribute subset {Age, Children} is a minimal reduct.

When a decision system, instead of an information system, is considered, the definition of a reduct B ($B \subseteq A$) can be modified as follows [25]:

$$\forall x_i, x_j \in U, \ \text{if } f_B(x_i) = f_B(x_j), \ \text{then } d(x_i) = d(x_j),$$

where $d(x_i)$ denotes the value of the decision attribute of the object x_i, $f_B(x_i)$ denotes the attribute values of x_i for the attribute set B. Similarly, if no subset of B can satisfy the above condition, B is called a minimal reduct in the decision system. Take the simple decision system shown in Table 2 as an example. It is modified from Table 1.

In Table 2, a decision attribute, Buying computers, is added to the original information system (Table 1) to form a decision system. In this example, the attribute subset {Age, Income} is not a reduct since the two objects x_1 and x_4 have the same values for the two attributes but belong to different classes. On the contrary, the attribute subset {Age, Children} is a reduct for the decision system. Furthermore, it is a minimal reduct since neither {Age} nor {Children} is a reduct. Finding minimal reducts has been proven as an NP-Hard problem. Li et al. [11] proposed the concept of "approximate" reducts to speed up the searching process. An approximate reduct allows for some reasonable tolerance degrees, but can greatly reduce the computation complexity. Next, the concept of relative dependency is introduced.

Table 1 A simple information system

Object	Age	Income	Children
x_1	Young	Low	No
x_2	Middle	Middle	Yes
x_3	Senior	High	Yes
x_4	Young	Low	Yes
x_5	Senior	Middle	No

Table 2 A simple decision system

Object	Age	Income	Children	Buying computers
x_1	Young	Low	No	No
x_2	Middle	Middle	Yes	No
x_3	Senior	High	Yes	Yes
x_4	Young	Low	Yes	Yes
x_5	Senior	Middle	No	No

2.2 Relative dependency

Han [5] and Li et al. [11] developed an approach based on the relative dependency to find approximate reducts. The relative dependency is motivated by the operation "projection", which is very important in the relational algebra. It can also be easily executed by SQL or other query languages. Given an attribute subset $B \subseteq A$ and a decision attribute d, the projection of the object set U on B is denoted by $\Pi_B(U)$ and can be computed by the following two steps: removing attributes in the different set ($A - B$) and merging all the remaining rows which are indiscernible [11]. Thus, among the tuples with the same attribute values for B, only one is kept and the others are removed. For example, the projection of the data in Table 2 on the attribute {Age} is shown below:

$$\Pi_{\{\text{Age}\}}(\text{U}) = \{x_1, x_2, x_3\}.$$

In this example, x_4 and x_5 are removed since they have the same value of the attribute Age as x_1 and x_3 have. Similarly, the projection on the attribute Children and on the attribute subset {Age, Children} is shown below:

$$\Pi_{\{\text{Children}\}}(U) = \{x_1, x_2\}, \quad \text{and}$$
$$\Pi_{\{\text{Age,Children}\}}(U) = \{x_1, x_2, x_3, x_4, x_5\}.$$

Han et al. thus defined the relative dependency degree (δ_B^D) of the attribute subset B with regard to the set of decision attributes D as follows:

$$\delta_B^D = \frac{|\Pi_B(U)|}{|\Pi_{B \cup D}(U)|},$$

where $|\Pi_B(U)|$ and $|\Pi_{B \cup D}(U)|$ are the numbers of tuples after the projection operations are performed on U according to B and $B \cup D$, respectively. Take the decision system shown in Table 2 as an example. $|\Pi_{\{\text{Age}\}}(U)| = |\{x_1, x_2, x_3\}| = 3$ and $|\Pi_{\{\text{Age, Buying computers}\}}(U)| = |\{x_1, x_2, x_3, x_4, x_3\}| = 5$. The relative dependency degree of {Age} with regard to {Buying computers} is thus 3/5, which is 0.6.

The goal of the paper is to cluster attributes such that the process of finding approximate reducts can be improved. For achieving this goal, it is thus important to develop an evaluation method which can measure the similarity of attributes.

This paper extends the concept of the relative dependency to compute the similarity between any two attributes and proposes an attribute clustering method. The proposed approach will be described in Sect. 3.

2.3 The *k*-means and the *k*-medoids clustering approaches

The *k*-means and the *k*-medoids approaches are two well-known partitioning (or clustering) strategies. They are widely used to cluster data when the number of clusters is given in advance. The *k*-means clustering approach [12] consists of two major steps: (1) reassigning objects to clusters and (2) updating the centers of clusters. The first step calculates the distances between each object and the k centers and reassigns the object to the group with the nearest center. The second step then calculates the new means of the k groups just updated and uses them as the new centers. These two steps are then iteratively executed until the clusters no longer change.

The *k*-medoids approach [8] adopts a quite different way of finding the centers of clusters. Assume k centers have been found. The *k*-medoids approach selects another object at random and replaces one of the original centers with the new object if better clustering results can be obtained. The absolute-error criterion [6] shown below is used to decide whether the replacement is better or not:

$$E = \sum_{j=1}^{k} \sum_{p \in C_j} |p - o_j|,$$

where E is the sum of the absolute errors for all the objects in the data set, p is an object in cluster C_j, o_j is the current center of C_j, and the absolute value $|p - o_j|$ means the distance between the two objects p and o_j. For each randomly selected object $o_{j'}$, one of the original k centers, say o_j, will be replaced with it and its new sum E' of absolute errors will be calculated. E' will then be compared with the previous E. If E' is less than E, then $o_{j'}$ is more suitable as a center than o_j. $o_{j'}$ thus actually replaces o_j as a new cluster center; otherwise, the replacement is aborted. The same procedure is repeated until the cluster centers no longer change.

The complexity of the *k*-medoids approach is in general higher than the *k*-means approach, but the former can guarantee that all the centers of clusters obtained are objects themselves. This feature is important to the proposed attribute clustering here, since not only the attributes are clustered but also the representative attribute of each cluster has to be found. On the contrary, the *k*-means approach may use non-object points as cluster centers. Note that both the *k*-means and the *k*-medoids approaches are mainly designed to cluster objects, but not attributes. As mentioned above, the goal of the paper is to cluster attributes. An attribute clustering method based on *k*-medoids is thus proposed to achieve this purpose. It also uses a better search strategy to find centers in a dense region, instead of random selection in *k*-medoids. Besides, a method to measure the distances (dissimilarities) among attributes is also needed.

3 Attribute dissimilarity

In this paper, we partition the attributes into k clusters according to the dependency between each pair of attributes. Each cluster can thus be represented by its representative attribute. The whole feature spaces can thus be greatly reduced.

For most clustering approaches, the distance between two objects is usually adopted as a measure for representing their dissimilarity, which is then used for deciding whether the objects belongs to the same cluster or not. In this paper, the attributes, instead of the objects, are to be clustered. The conventional distance measures such as Euclidean distance or Manhattan distance are thus not suitable since the attributes may have different formats of data, which are hard to compare. For example, assume there are two attributes, one of which is age and the other is gender. It is thus hard to compare the two attributes via the traditional distance measure. Below, a measure based on the concept of relative data dependency is proposed to achieve it. It was proposed by Han et al. [5] and can be thought of as a kind of similarity degrees.

Given two attributes A_i and A_j, the relative dependency degree of A_i with regard to A_j is denoted by $\mathrm{Dep}(A_i, A_j)$ and is defined as:

$$\mathrm{Dep}(A_i,\ A_j) = \frac{\left|\Pi_{A_i}(U)\right|}{\left|\Pi_{A_i,\ A_j}(U)\right|},$$

where $\left|\Pi_{A_i}(U)\right|$ is the projection of U on attribute A_i. Note that the original relative dependency degree only considers the relative dependency between a condition attribute set and a decision attribute set. Here we extend the above formula to estimate the relative data dependency between any pair of attributes. The dependency degree is not symmetric, such that the condition $\mathrm{Dep}(A_i, A_j) = \mathrm{Dep}(A_j, A_i)$ is not always valid. We thus use the average of $\mathrm{Dep}(A_i, A_j)$ and $\mathrm{Dep}(A_j, A_i)$ to represent the similarity of the two attributes A_i and A_j. This extended relative dependency is thought of as the similarity of the two attributes. The distance (dissimilarity) measure for the pair of attributes A_i and A_j is thus proposed as follows:

$$d(A_i,\ A_j) = \frac{1}{\mathrm{Avg}(\mathrm{Dep}(A_i,\ A_j),\ \mathrm{Dep}(A_j,\ A_i))}.$$

Take the distance between the two attributes Age and Children in Table 2 as an example. Since $|\Pi_{\{\mathrm{Age}\}}(U)| = 3$, $|\Pi_{\{\mathrm{Children}\}}(U)| = 2$ and $|\Pi_{\{\mathrm{Age,Children}\}}(U)| = 5$, the relative dependency degrees Dep(Age, Children) and Dep (Children, Age) are 0.6 and 0.4, respectively. The distance d(Age, Children) is thus 1/Avg(0.6, 0.4), which is 2.

4 The proposed algorithm

In this section, an attribute clustering algorithm called Most Neighbors First (MNF) is proposed to cluster the attributes into a fixed number of groups. Assume the number k of desired clusters is known. Some preprocessing steps such as removal of inconsistent or incomplete tuples and discretization of numerical data are first done. After that, the proposed MNF attribute clustering algorithm is used to partition the feature space into k clusters and output the k representative attributes of the clusters.

The proposed clustering algorithm MNF is based on the k-medoids approach. Unlike the k-means approach, the proposed algorithm always updates the centers by some existing objects. Besides, it uses a better search strategy to find centers in a dense region, instead of random selection in k-medoids.

The proposed algorithm MNF consists of two major phases: (1) reassigning the attributes to the clusters and (2) updating the centers of the clusters. In the first phase, the proposed distance measure is used to find the nearest center of each attribute. The attribute is then assigned to the cluster with that center. In the second phase, each cluster C_i uses a searching radius r_i to decide the neighbors of each attribute in C_i. The attribute with the most neighbors in a cluster is then chosen as the new center. The proposed algorithm is described in details below.

The MNF attribute clustering algorithm:

Input: An information system $I = (U, A \cup \{d\})$ and the number k of desired clusters.

Output: k appropriate attribute clusters with their representative attributes.

Step 1 Randomly select k attributes $\{A_1^c, A_2^c, \ldots, A_k^c\}$ as the initial representative attributes (centers) in the k clusters, where A_t^c stands for the representative attribute (center) of the t-th cluster C_t, $A_t^c \in A$. Denote $A_c = \{A_1^c, A_2^c, \ldots, A_k^c\} \subseteq A$ as the initial representative attribute set.

Step 2 For each non-representative attribute $A_i \in A - A_c$, compute the dissimilarity (distance) $d(A_i, A_t^c)$ between attribute A_i and each representative attribute A_t^c as:

$$d(A_i, A_t^c) = \frac{1}{\text{Avg}(\text{Dep}(A_i, A_t^c), \text{Dep}(A_t^c, A_i))},$$

where $\text{Dep}(A_i, A_t^c)$ represents the relative dependency degree of A_i with regard to A_t^c and $\text{Dep}(A_t^c, A_i)$ represents the relative dependency degree of A_t^c with regard to A_i, $t \in \{1, 2, \ldots, k\}$.

Step 3 Allocate all non-center attributes to their nearest centers according to the distances found in Step 2. Collect a center attribute with its allocated attributes as a cluster.

Step 4 For each cluster C_t, calculate the distances between any two different attributes within C_t.

Step 5 Calculate the radius r_t of each cluster C_t as:

$$r_t = \frac{\sum_{i \neq j} d(A_{t,i}, A_{t,j})}{C_2^{n_t}},$$

where $d(A_{t,i}, A_{t,j})$ is the distance between any two attributes $A_{t,i}$ and $A_{t,j}$ within the cluster C_t, n_t is the number of attributes within C_t, and $C_2^{n_t}$ is the number of attribute pairs in the cluster, which is $\frac{n_t(n_t-1)}{2}$.

Step 6 For each attribute $A_{t,i}$ (including the center A_t^c) within a cluster C_t, find the set of attributes [(called $\text{Near}(A_{t,i})$] with their distances from $A_{t,i}$ within r_t. That is:

$$\text{Near}(A_{t,i}) = \{A_{t,j} \mid A_{t,j} \in C_t \text{ and } d(A_{t,i}, A_{t,j}) \leq r_t\}.$$

Step 7 For each cluster C_t, find the attribute $A_{t,l}$ with the most attributes in its Near set. Set $A_{t,l}$ as the new center A_t^c of C_t.

Step 8 Repeat Steps 2–7 until the clusters have converged.

Step 9 Output the final clusters and their centers as the representative attributes.

After Step 9, k clusters of attributes are formed and k representative attributes for the feature space are found.

5 An example

In this section, a simple example is given to show how the proposed algorithm can be used to cluster the attributes. Table 3 shows the scores of eight students. There are eight condition attributes A = {PR, CA, DM, C++, JAVA, DB, DS, AL}, respectively stands for the eight subjects: Probability, Calculus, Discrete Mathematics, C++, JAVA, Database, Data Structure and Algorithms. The values of the condition attributes are $\{A, B, C, D\}$, which stand for the grade levels of a subject. There is one decision attribute {ST}, which stands for {Study for Master Degree} and has two possible classes {Yes, No}. In this example, the number of clusters is

Table 3 An example for attribute clustering

Object	PR	CA	DM	C++	JAVA	DB	DS	AL	ST
x_1	*A*	*B*	*A*	*B*	*B*	*A*	*B*	*B*	Yes
x_2	*A*	*B*	*B*	*C*	*A*	*B*	*C*	*B*	No
x_3	*B*	*B*	*B*	*A*	*B*	*B*	*A*	*A*	Yes
x_4	*B*	*C*	*C*	*C*	*C*	*B*	*C*	*C*	No
x_5	*C*	*C*	*C*	*D*	*C*	*C*	*D*	*C*	No
x_6	*B*	*B*	*C*	*D*	*C*	*D*	*D*	*C*	No
x_7	*B*	*B*	*C*	*B*	*B*	*A*	*B*	*C*	Yes
x_8	*A*	*A*	*A*	*A*	*B*	*B*	*A*	*B*	Yes

set at 2 (i.e. $k = 2$). For the set of data, the proposed algorithm proceeds as follows.

Step 1 k attributes are randomly selected as the initial centers of the clusters. In this example, k is set at 2. Assume that the two attributes DM and DS are selected as the initial centers of the two clusters C_1 and C_2, respectively.

Step 2 The distances (dissimilarities) between each non-center attribute and each center are calculated. Take the distance between PR and DM as an example. Since $|\Pi_{PR}| = 3$, $|\Pi_{DM}| = 3$ and $|\Pi_{PR,DM}| = 5$, the relative dependency degrees Dep(PR, DM) is calculated as 0.6 and Dep(DM, PR) is 0.6 as well. The distance between the two attributes is thus calculated as:

$$d(\text{PR, DM}) = \frac{1}{\text{Avg}(0.6, 0.6)} = 1.67.$$

All the distances between non-center attributes and representative centers are shown in Table 4.

Step 3 All non-center attributes are allocated to their nearest centers. Thus, cluster C_1 contains {PR, CA, AL, DM} and cluster C_2 contains {C++, JAVA, DB, DS}.

Step 4 The distances between any two different attributes in the same clusters are calculated. The results are shown in Table 5.

Step 5 The searching radius of each cluster is calculated. Take the cluster C_1 as an example. It includes four attributes {PR, CA, AL, DM}. The distances between each pair of attributes in C_1 are {1.67, 1.67, 1.33, 1.67, 1.67, 1.33}. The radius r_1 is then calculated as:

$$r_1 = \frac{1.67 + 1.67 + 1.33 + 1.67 + 1.67 + 1.33}{6} = 1.56.$$

Step 6 The Near set of each attribute in a cluster is calculated. Take the attribute PR in cluster C_1 as an example. Its distance from the other three attributes CA, AL and DM in the same cluster are calculated as 1.67, 1.33 and 1.67. Near(PR) thus includes only the attribute AL since only AL is within the radius r_1 (1.56), which is found from Step 5. Similarly, the Near sets of the other three attributes in the cluster C_1 are found as follows:

$$\begin{aligned} &\text{Near}(CA) = \phi, \\ &\text{Near}(AL) = \{\text{PR, DM}\}, \text{ and} \\ &\text{Near}(DM) = \{\text{AL}\}. \end{aligned}$$

Step 7 Since the attribute AL has the most attributes in its Near set for the cluster C_1, AL then replaces the attribute DM as the new center of C_1. Similarly, the original center DS for C_2 has the most attributes in its Near set. DS is thus still the center of C_2.

Step 8 Steps 2–7 are repeated until the two clusters no longer change. The final clusters can thus be found as follows:
$C_1 = \{\text{PR, CA, AL, DM}\}$, with the center AL.
$C_2 = \{\text{C++, JAVA, DB, DS}\}$, with the center DS.

Step 9 The final clusters and their centers as the representative attributes are then output. The attributes in the same cluster can thus be considered to possess similar characteristics in classification and can be used as alternative attributes of the representative one.

Table 4 The distances between non-center attributes and representative centers

Cluster C_1		Cluster C_2	
Attribute pair	Distance	Attribute pair	Distance
d(PR, DM)	1.67	d(PR, DS)	2.33
d(CA, DM)	1.67	d(CA, DS)	2.27
d(C++, DM)	2	d(C++, DS)	1
d(JAVA, DM)	1.67	d(JAVA, DS)	0.8
d(DB, DM)	2	d(DB, DS)	0.8
d(AL, DM)	1.33	d(AL, DS)	2

Table 5 The distances between any two attributes within the same clusters

Within cluster C_1		Within cluster C_2	
Attribute pair	Distance	Attribute pair	Distance
d(PR, DM)	1.67	d(C++, DS)	1
d(PR, AL)	1.33	d(C++, DB)	1.25
d(CA, AL)	1.67	d(JAVA, DB)	2
d(PR, CA)	1.67	d(C++, JAVA)	1.67
d(CA, DM)	1.67	d(JAVA, DS)	1.25
d(AL, DM)	1.33	d(DB, DS)	1.25

6 Experimental results

In this section, the implementation of the proposed algorithm for clustering attributes is described. The experiments were implemented in C++ on an AMD Athlon 64 X2 Dual Core 3800+ personal computer with 2.01 GHz and 1 GB RAM. The real-world dataset, the Wisconsin Database of Breast Cancers (WDBC) [24], was used to verify the approach. The characteristics of the dataset are shown in Table 6.

Each attribute in the dataset is numerical, so discretization should be first done. In this paper, the discretization is performed by two methods. The first one is equal width which

Table 6 The characteristics of the dataset of WDBC

Number of instances	569
Number of attributes	30
Number of classes	2
Number of missing attributes	0

discretizes the range of an attribute in equal intervals; the other one is equal frequency which considers the appearing frequency of each value of an attribute. The average intradistance (dissimilarity) in a cluster is used as a measure to evaluate the goodness of the results. It is defined as the average distance between an attribute and its representative attribute in the same cluster. Formally, it can be represented by the following formula:

$$\text{AvgIntraD} = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{|C_i|-1}\sum_{A_j \in C_i - A_i^c} d(A_j,\ A_i^c),$$

where C_i is the i-th attribute cluster. The results of AvgIntra D along with different cluster numbers by the two discretization methods are shown in Fig. 1.

As Fig. 1 showed, the average intradistances decreased along with the increase of the cluster number for both the two discretization methods. Besides, the discretization method by equal width performed better than that by equal frequency.

Another evaluation measure for the clustering results was by the average intrasimilarity in clusters, which was defined as follows:

$$\text{AvgIntraS} = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{|C_i|-1}\sum_{A_j \in C_i - A_i^c} \text{Sim}(A_j,\ A_i^c),$$

where $\text{Sim}(A_j, A_i^c)$ denotes the average dependency degree for a non-representative attribute A_j and its representative attribute in the same class. $\text{Sim}(A_j, A_i^c)$ was computed as follows:

$$\text{Sim}(A_j,\ A_i^c) = \frac{\text{Dep}(A_j,\ A_i^c) + \text{Dep}(A_i^c,\ A_j)}{2}.$$

The results of AvgIntra S along with different cluster numbers by the two discretization methods are shown in Fig. 2.

As Fig. 2 showed, the average intrasimilarity increased along with the increase of the cluster number for both the two discretization methods. The same as before, the discretization method by equal width performed better than that by equal frequency.

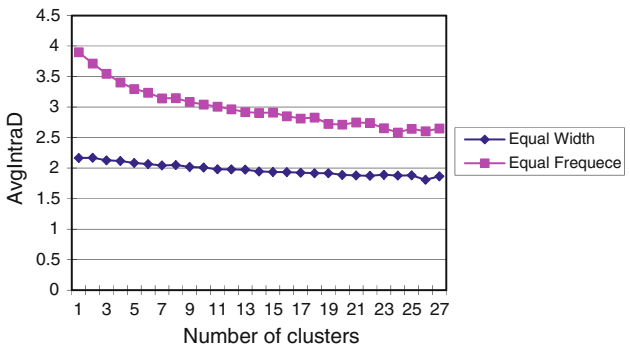

Fig. 1 The average intra distances along with different cluster numbers by the two discretization methods

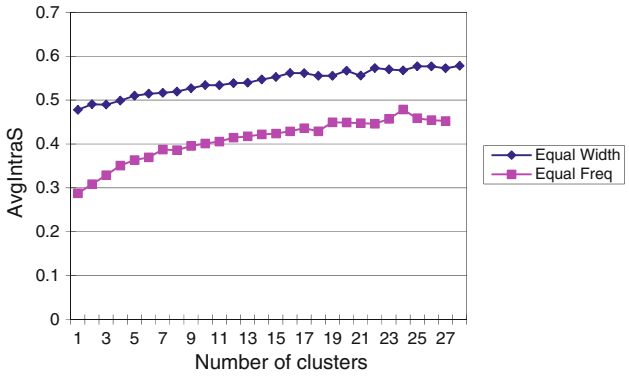

Fig. 2 The average intra similarities along with different cluster numbers by the two discretization methods

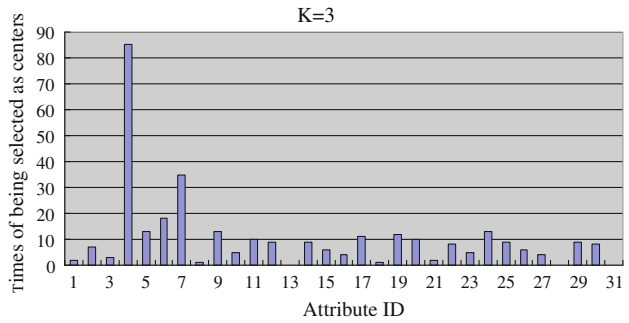

Fig. 3 The frequencies of being selected as centers for $k = 3$

The main purpose of attribute clustering was to select representative attributes to replace the whole set. Since the representative attributes selected by the algorithms were not always the same, the algorithm was thus run for 100 times and the frequency for each attribute being selected as a representative attribute was counted. For example, the selected frequencies of all the attributes when the cluster number k is 3 are shown in Fig. 3.

As Fig. 3 showed, attribute 4 and attribute 7 were the two most frequently chosen attributes in the experiments. Therefore, the two attributes could be chosen and another one might be selected from the set of attributes 5, 6, 9, 17, 19, 24, which had their frequencies more than 10 times and formed the third cluster. The results for $k = 6$ and $k = 9$ are also shown in Figs. 4, 5 for a comparison.

As Figs. 4 and 5 showed, the difference of the frequencies of the attributes being selected as centers was smaller and smaller when the cluster number k increased. This phenomenon resulted from the fact that the attributes in the same cluster would become more similar to each other when the cluster number increased. Attribute would thus be chosen as centers with a more uniform opportunity. In this case, some other criteria, such as attribute cost and ratio of missing values may be used to aid the selection of representative attributes.

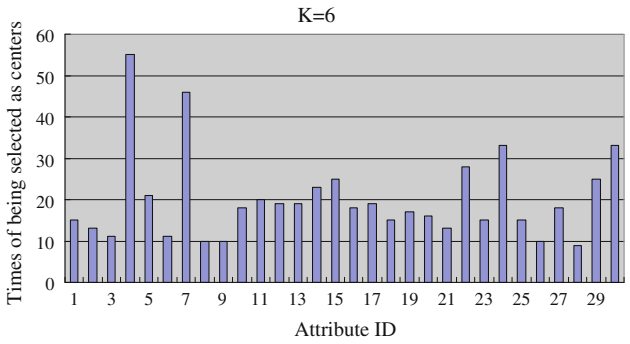

Fig. 4 The frequencies of being selected as centers for $k = 6$

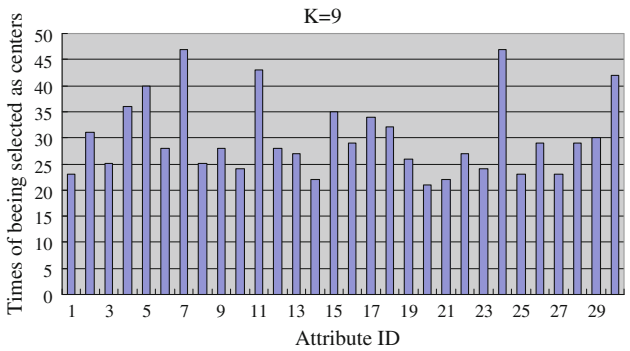

Fig. 5 The frequencies of being selected as centers for $k = 9$

7 Conclusions and future work

In this paper, we have attempted to use attribute clustering for feature selection. A measure of the attribute dissimilarity based on the relative dependency is proposed to calculate the distance between two attributes. An attribute clustering algorithm, called Most Neighbors First, has also been proposed to find centers in a dense region, instead of random selection in k-medoids. The proposed attribute clustering approach consists of two major phases: reassigning attributes to clusters and updating centers of clusters. After the attributes are organized into several clusters by their similarity degrees, the representative attributes in the clusters can be used for classification such that the whole feature space can be greatly reduced. Besides, if the values of some representative attributes cannot be obtained from current environments for inference, some other possible attributes in the same clusters can be used to achieve approximate inference results.

Experimental results show that the average similarity in the same cluster will increase along with the increase of cluster numbers. Besides, the discretization method is an important factor for the final results. The discretization method by equal width performs better than that by equal frequency.

At last, the proposed attribute clustering approach has to know the number of clusters in advance. This requirement results in the limitation of its applications. In the future, we will try to develop other new approaches for attribute clustering, while the number of clusters is unknown. We will also attempt to apply the proposed approach to some real application domains.

References

1. Al-Radaideh, Q.A., Sulaiman, M.N., Selamat, M.H., Ibrahim, H.: Approximate reduct computation by rough sets based attribute weighting. In: The IEEE International Conference on Granular, Computing, vol. 2, pp. 383–386 (2005)
2. Buchanan, B.G., Shortliffe, E.H.: Rule-based expert system: the MYCIN experiments of the Standford heuristic programming projects. Addison-Wesley, Massachusetts (1984)
3. Dong, J.Z., Zhong, N., Ohsuga, S.: Using rough sets with heuristics to feature selection, new directions in rough sets data mining, granular-soft computing. Springer, Berlin (1999)
4. Gao, K., Liu, M., Chen, K., Zhou, N., Chen, J.: Sampling-based tasks scheduling in dynamic grid environment. In: The Fifth WSEAS International Conference on Simulation, Modeling and Optimization, pp. 25–30 (2005)
5. Han, J.: Feature selection based on rough set and information entropy. In: The IEEE International Conference on Granular Computing, vol. 1, 153–158 (2005)
6. Han, J., Kamber, M.: Data mining: concepts and techniques. Morgan Kaufmann, San Francisco (2006)
7. Hong, T.P., Wang, T.T., Wang, S.L.: Knowledge acquisition from quantitative data using the rough-set theory. Intell Data Anal. **4**, 289–304 (2000)
8. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis. John Wiley & Sons, Toronto (1990)
9. Kodratoff, Y., Michalski, R.S.: Machine learning: an artificial intelligence artificial intelligence approach. Morgan Kaufmann Publishers, San Mateo (1983)
10. Komorowski, J., Polkowski, L., Skowron, A.: Rough sets: a tutorial. http://www.let.uu.nl/esslli/Courses/skowron/skowron.ps
11. Li, Y., Shiu, S.C.K., Pal, S.K.: Combining feature reduction and case selection in building CBR classifiers. IEEE Trans. Knowl. Data Eng. **18**(3), 415–429 (2006)
12. Lloyd, S.P.: Least square quantization in PCM. Bell Labs, USA (1957)
13. Mamat, R., Herawan, T., Deris, M.M.: MAR: maximum attribute relative of soft set for clustering attribute selection. Knowl. Based Syst. **52**, 11–20 (2012)
14. Michalski, R.S., Carbonell, J.G., Mitchell, T.M.: Machine learning: an artificial intelligence approach. Morgan Kaufmann Publishers, Los Altos (1983)
15. Michalski, R.S., Carbonell, J.G., Mitchell, T.M.: Machine learning: an artificial intelligence approach. Morgan Kaufmann Publishers, Los Altos (1983)
16. Parmar, D., Wu, T., Blackhurst, J.: MMR: an algorithm for clustering categorical data using rough set theory. Data Knowl. Discov. **63**(3), 879–893 (2007)
17. Pawlak, Z.: Rough set. Int J Comput Inf Sci **11**(5), 341–356 (1982)
18. Pawlak, Z.: Why rough sets? In: The Fifth IEEE International Conference on Fuzzy Systems, vol. 2, pp. 738–743 (1996)
19. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Int. J. Comput. Inf. Sci. **177**(1), 3–27 (2007)
20. Qin, H., Ma, X., Zain, J.M., Herawan, T.: A novel soft set approach in selecting clustering attribute. Knowl. Based Syst. **36**, 139–145 (2012)
21. Riley, G.: Expert systems: principles and programming. PWS-Kent, Boston (1989)

22. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems, Handbook of Application and Advances of the Rough Sets Theory, pp. 331–362. Kluwer Academic Publishers, Dordrecht (1992)
23. Sun, H.Q., Xiong, Z.: Finding minimal reducts from incomplete information systems. In: The Second International Conference on Machine Learning and Cybernetics, vol. 1, pp. 350–354 (2003)
24. Wolberg, W.H., Street W.N., Mangasarian O.L.: (1995), UCI machine learning repository, http://www.ics.uci.edu/mlearn/MLRepository.html, Irvine, CA: University of California, Department of Information and Computer Science
25. Wroblewski, J.: Finding minimal reducts using genetic algorithms. In: The Second Annual Join Conference on Information Sciences, pp. 186–189 (1995)
26. Zhang, J., Wang, J., Li, D., He, H., Sun, J.: A new heuristic reduct algorithm based on rough sets theory. Lecture Notes in Computer Science, pp. 247–253. Springer, New York (2003)
27. Zhang, M., Yao, J.T.: A rough sets based approach to feature selection. In: The IEEE Annual Meeting of Fuzzy, Information, pp. 434–439 (2004)