Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Data & Knowledge Engineering xxx (2004) xxx–xxx
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Categorical data visualization and clustering using subjective factors

Chia-Hui Chang *, Zhi-Kai Ding

Department of Computer Science and Information Engineering, National Central University, No. 300,
Jhungda Road, Jhungli City, Taoyuan 320, Taiwan

Received 3 April 2004; accepted 1 September 2004

Abstract

Clustering is an important data mining problem. However, most earlier work on clustering focused on numeric attributes which have a natural ordering to their attribute values. Recently, clustering data with categorical attributes, whose attribute values do not have a natural ordering, has received more attention. A common issue in cluster analysis is that there is no single correct answer to the number of clusters, since cluster analysis involves human subjective judgement. Interactive visualization is one of the methods where users can decide a proper clustering parameters. In this paper, a new clustering approach called CDCS (Categorical Data Clustering with Subjective factors) is introduced, where a visualization tool for clustered categorical data is developed such that the result of adjusting parameters is instantly reflected. The experiment shows that CDCS generates high quality clusters compared to other typical algorithms.

Keywords: Data mining; Cluster analysis; Categorical data; Cluster visualization

* Corresponding author. Fax: +886 3 4222681.
E-mail addresses: chia@csie.ncu.edu.tw (C.-H. Chang), sting@db.csie.ncu.edu.tw (Z.-K. Ding).

0169-023X/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.datak.2004.09.001

1. Introduction

Clustering is one of the most useful tasks for discovering groups and identifying interesting distributions and patterns in the underlying data. The clustering problem is about partitioning a given data set into groups (clusters) such that the data points in a cluster are more similar to each other than points in different clusters. The clusters thus discovered are then used for describing characteristics of the data set. Cluster analysis has been widely used in numerous applications, including pattern recognition, image processing, land planning [21], text query interface [11], market research, etc.

Many clustering methods have been proposed in the literature and most of these handle data sets with numeric attributes, where proximity measure can be defined by geometrical distance. For categorical data which has no order relationships, a general method is to transform it into binary data. However, such binary mapping may lose the meaning of original data set and result in incorrect clustering, as reported in [7]. Furthermore, high dimensions will require more space and time if the similarity function involves with matrix computation, such as the Mahalanobis measure [16].

Another problem we face in clustering is how to validate the clustering results and decide the optimal number of clusters that fits a data set. Most clustering algorithms require some predefined parameters for partitioning. These parameters influence the clustering result directly. For a specific application, it may be important to have well separated clusters, while for another it may be more important to consider the compactness of the clusters. Hence, there is no correct answer for the optimal number of clusters since cluster analysis may involve human subjective judgement, and visualization is one of the most intuitive ways for users to decide a proper clustering.

In Fig. 1 for example, there are 54 objects displayed. For some people, there are two clusters, while some may think there are six clusters, still others may think there are 18 clusters, depending on their subjective judgement. In other words, a value can be small in a macroscopic view, but it can be large in a microscopic view. The definition of similarity varies with respect to different views. Therefore, if categorical data can be visualized, parameter adjustment can be easily done, even if several parameters are involved.

In this paper, we present a method for visualization of the clustered categorical data such that users' subjective factors can be reflected by adjusting clustering parameters, and therefore to increase the clustering result's reliability. The proposed method, CDCS (Categorical Data Clustering using Subjective factors), can be divided into three steps: (1) the first step incorporates a single-pass clustering method to group objects with high similarity, (2) then, small clusters are merged

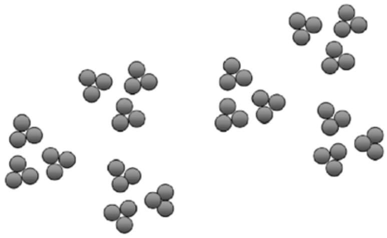


Fig. 1. Fifty-four objects displayed in a plane.

and displayed for visualization, and (3) through the proposed interactive visualization tool, users can observe the data set and determine appropriate parameters for clustering.

The rest of the paper is organized as follows. Section 2 reviews related work in categorical data clustering and data visualization. Section 3 introduces the architecture of CDCS and the clustering algorithm utilized. Section 4 discusses the visualization method of CDCS in detail. Section 5 presents an experimental evaluation of CDCS using popular data sets and comparisons with two famous algorithms AutoClass [2] and k-mode [9]. Section 6 presents the conclusions and suggests future work.

2. Related work

Clustering is broadly recognized as a useful tool for many applications. Researchers of many disciplines (such as databases, machine learning, pattern recognition, statistics, etc.) have addressed clustering problem in many ways. However, most researches concern numerical type data which has geometrical shape and clear distance definition, while little attention has been paid to categorical data clustering. Visualization is an interactive, reflective method that supports exploration of data sets by dynamically adjusting parameters to see how they affect the information being presented. In this section, we review works on categorical data clustering and visualization methods.

2.1. Categorical data clustering

In recent years, a number of clustering algorithms for categorical data have been proposed, partly due to increasing applications in market baskets analysis, customer databases, etc. We briefly review the main algorithms below.

One of the most common ways to solve categorical data clustering is to extend existing algorithms with a proximity measure for categorical data, and many clustering algorithms belong to this category. For example, k-mode [9] is based on k-mean [15] but adopts a new similarity function to handle categorical data. A cluster center for k-mode is represented by a virtual object with the most frequent attribute values in the cluster. Thus, the distance between two data objects is defined as the number of different attribute values.

ROCK [7] is a bottom-up clustering algorithm which adopts a similarity function based on the number of common neighbors, which is defined by the Jaccard coefficient [18]. Two data objects are more similar if they have more common neighbors. Since the time complexity for the bottom-up hierarchical algorithm is quadratic, it clusters a randomly sampled data set and then partitions the entire data set based on these clusters. COBWEB [3], on the other hand, is a top-down clustering algorithm which constructs a classification tree to record cluster information. The disadvantage of COBWEB is that its classification tree is not height-balanced for skewed input data, which may cause increased time and space cost.

AutoClass [2] is an EM-based approach which is supplemented by a Bayesian evaluation for determining the optimal classes. It assumes a predefined distribution for data and tries to maximize the function with appropriate parameters. AutoClass requires a range for the number of clusters as input. Because AutoClass is a typical iterative clustering algorithm, if the data cannot be entirely loaded into memory, the time cost will be expensive.

STIRR [6] is an iterative method based on non-linear dynamical systems. Instead of clustering objects themselves, the algorithm aims at clustering co-occur attribute values. Usually, this approach can discover the largest cluster with the most attribute values, and even if the idea of orthonormal is introduced [23], it can only discover one more cluster. CACTUS [5] is another algorithm that conducts clustering from attribute relationship. CACTUS employs a combination of inter-attribute and intra-attribute summaries to find clusters. However, there has been no report on how such an approach can be used for clustering general data sets.

Finally, several endeavors have been tried to mine clusters with association rules. For example, Kosters et al. proposed the clustering of a specific type of data sets where the objects are vectors of binary attributes from association rules [8]. The hypergraph-based clustering in [13] is used to partition items and then transactions based on frequent itemsets. Wang et al. [22] also focus on transaction clustering as in [8].

2.2. Visualization methods

Since human vision is endowed with the classification ability for graphic figures, it would greatly help solving the problem if the data could be graphically transformed for visualization. However, human vision is only useful for figures with low dimension. For high-dimensional data, it must be mapped into low dimensions for visualization, and there are several visualization methods including linear mapping and non-linear mapping. Linear mapping, like principle component analysis, is effective but cannot truly reflect the data structure. Non-linear mapping, like Sammon projection [12] and SOM [10] requires more computation but is better at preserving the data structure. However, whichever method is used, traditional visualization methods can transform only numerical data. For categorical data, visualization is only useful for attribute dependency analysis and is not helpful for clustering. For example, the mosaic display method [4], a popular statistical visualization method, displays the relationships between two attributes. Users can view the display in a rectangle composed of many mosaic graphs and compare it to another mosaic display, assuming that two attributes are independent. The tree map [19], which is the only visualization method which can be used for distribution analysis, transforms data distribution to a tree composed of many nodes. Each node in this tree is displayed as a rectangle and its size represents the frequency of an attribute value. Users can thus observe an overview of the attribute distribution. However, it provides no insight to clustering. Finally, reordering of attribute values may help visualization for categorical data with one attribute as proposed in [14].

3. Categorical data clustering and visualization

In this section, we introduce the clustering and visualization approach in our framework. We utilize the concept of Bayesian classifiers as a proximity measure for categorical data. The process of CDCS can be divided into three steps: (1) in the first step, it applies "simple clustering seeking" [20] to group objects with high similarity, (2) then, small clusters are merged and displayed by categorical cluster visualization, (3) users can then adjust the merging parameters and view the result

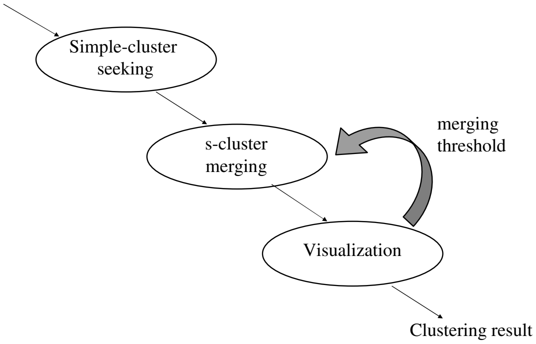


Fig. 2. The CDCS process.

through the interactive visualization tool. The process continues until users are satisfied with the result (see Fig. 2).

Simple cluster seeking, sometimes called dynamic clustering, is a one pass clustering algorithm which does not require the specification of the desired number of clusters. Instead, a similarity threshold is used to decide if a data object should be grouped into an existing cluster or form a new cluster. More specifically, the data objects are processed individually and sequentially. The first data object forms a single cluster by itself. Next, each data object is compared to existing clusters. If its similarity with the most similar cluster is greater than a given threshold, this data object is assigned to that cluster and the representation of that cluster is updated. Otherwise, a new cluster is formed. The advantage of dynamic clustering is that it provides simple and incremental clustering where each data sample contributes to changes in the clusters. Besides, the time complexity, O(kn), for clustering n objects into k clusters is suitable for handling large data sets.

However, there is one inherent problem for this dynamic clustering: the clustering result can be affected by the input order of data objects. To solve this problem, higher similarity thresholds can be used to decrease the influence of the data order and ensure that only highly similar data objects are grouped together. As a large number of small clusters (called s-clusters) can be produced, the cluster merging step is required to group s-clusters into larger groups. Therefore, a merging step similarity threshold is designed to be adjusted for interactive visualization. Thus, a user's views about the clustering result can be extracted when he/she decides a proper threshold.

3.1. Proximity measure for categorical data

Clustering is commonly known as an unsupervised learning process. The simple cluster seeking approach can be viewed as a classification problem since it predicts whether a data object belongs to an existing cluster or class. In other words, data in the same cluster can be considered as having the same class label. Therefore, the similarity function of a data object to a cluster can be represented by the probability that the data object belongs to that cluster. Here, we adopt a similarity function based on the naive Bayesian classifier [17], which is used to compute the largest posteriori probability $\mathrm{Max}_j P(C_j|X)$ for a data object $X = (v_1, v_2, \ldots, v_d)$ to an existing cluster C_j. Using Bayes' theorem, $P(C_j|X)$ can be computed by

$$P(C_j|X) \propto P(X|C_j)P(C_j) \tag{1}$$

Assuming attributes are conditionally independent, we can replace $P(X|C_j)$ by $\prod_{i=1}^{d} P(v_i|C_j)$, where v_i is X's attribute value for the ith attribute. $P(v_i|C_j)$, a simpler form for $P(A_i = v_i|C_j)$, is the probability of v_i for the ith attribute in cluster C_j, and $P(C_j)$ is the priori probability defined as the number of objects in C_j to the total number of objects observed.

Applying this idea in dynamic clustering, the proximity measure of an incoming object X_i to an existing cluster C_j can be computed as described above, where the prior objects $X_1, \ldots, X_{i-1}$ before X_i are considered as the training set and objects in the same cluster are regarded as having the same class label. For the cluster C_k with the largest posteriori probability, if the similarity is greater than a threshold g defined as

$$g = p^{d-e} \times \epsilon^{e} \times P(C_k) \tag{2}$$

then X_i is assigned to cluster C_k and $P(v_i|C_k)$, $i = 1, \ldots, d$, are updated accordingly. For each cluster, a table is maintained to record the pairs of attribute value and their frequency for each attribute. Therefore, to update $P(v_i|C_k)$ is simply an increase of the frequency count. Note that to avoid zero product, the probability $P(v_i|C_j)$ is computed by $\frac{N_j(v_i)+m\cdot r}{|C_j|+m}$, where $N_j(v_i)$ is the number of examples in cluster C_j having attribute value v_i, r is the reciprocal of the number of values for the ith attribute as suggested in [17].

The equation for the similarity threshold is similar to the posteriori probability $P(C_j|X) = \prod_{i=1}^{d} P(v_i|C_j)P(C_j)$, where the symbol p denotes the average proportion of the highest attribute value for each attribute, and e denotes the number of attributes that can be allowed/tolerated for various values. For such attributes, the highest proportion of different attribute values is given a small value ϵ. This is based on the idea that the objects in the same cluster should possess the same attribute values for most attributes, while some attributes may be quite dissimilar. In a way, this step can be considered as density based clustering since probabilistic proximity measure is the basis in the density based clustering. For large p and small e, we will have many compact s-clusters. In the most extreme situation, where $p = 1$ and $e = 0$, each distinct object is classified to a cluster. CDCS adopts a default value 0.9 and 1 for p and e, respectively. The resulting clusters are usually small, highly condensed and applicable for most data sets.

3.2. Group merging

In the second step, we group the resulting s-clusters from dynamic clustering into larger clusters ready for display with the proposed visualization tool. To merge s-clusters, we first compute the similarity scores for each cluster pair. The similarity score between two s-clusters C_x and C_y is defined as follows:

$$\mathrm{sim}(C_x, C_y) = \prod_{i=1}^{d} \left[\sum_{j}^{|A_i|} \min\{P(v_{i,j}|C_x), P(v_{i,j}|C_y)\} + \epsilon \right] \tag{3}$$

where $P(v_{i,j}|C_x)$ denotes the probability of the jth attribute value for the ith attribute in cluster C_x, and $|A_i|$ denotes the number of attribute values for the ith attribute. The idea behind this definition is that the more the clusters intersect, the more similar they are. If the distribution of attribute

	1	2	3	4	5	6
1	1	0	0	0	1	0
2	0	1	0	0	0	0
3	0	0	1	1	0	0
4	0	0	1	1	0	0
5	1	0	0	0	1	1
6	0	0	0	0	1	1

Resulting groups:
{1,5,6},
{2},
{3,4}

Fig. 3. Binary similarity matrix (BM).

values for two clusters is similar, they will have a higher similarity score. There is also a merge threshold g', which is defined as follows:

$$g' = (p')^{d-e'} \times \epsilon^{e'} \tag{4}$$

Similar to the last section, the similarity threshold g' is defined by p', the average percentage of common attribute values for an attribute; and e', the number of attributes that can be allowed/tolerated for various values. The small value ϵ is given to be the reciprocal of the number of samples in the data set.

For each cluster pair C_x and C_y, the similarity score is computed and recorded in a $n \times n$ matrix SM, where n is the number of s-clusters. Given the matrix SM and a similarity threshold g', we compute a binary matrix BM (of size $n \times n$) as follows. If SM[x, y] is greater than the similarity threshold g', cluster C_x and C_y are considered similar and BM[x, y] = 1. Otherwise, they are dissimilar and BM[x, y] = 0. Note that the similarity matrix, SM, is computed only once after the single-pass clustering. For each parameter adjustment (g') by the user, the binary matrix BM is computed without recomputing SM. Unless the parameter for the first step is changed, there is no need to recompute SM.

With the binary matrix BM, we then apply a transitive concept to group s-clusters. To illustrate this, in Fig. 3, clusters 1, 5, and 6 can be grouped in one cluster since clusters 1 and 5 are similar, and clusters 5 and 6 are also similar (the other two clusters are {2} and {3, 4}). This merging step requires O(n^2) computation, which is similar to hierarchical clustering. However, the computation is conducted for n s-clusters instead of data objects. In addition, this transitive concept allows arbitrarily shaped clusters to be discovered.

4. Visualization with CDCS

Simply speaking, visualization in CDCS is implemented by transforming a cluster into a graphic line connected by 3D points. These three dimensions represent the attributes, attribute values and the percentages of an attribute value in the cluster. These lines can then be observed in 3D space through rotations to see if they are close to each other. In the following, we first introduce the principle behind our visualization method; and then describe how it can help determine a proper clustering.

4.1. Principle of visualization

Ideally, each attribute A_i of a cluster C_x has an obvious attribute value $v_{i,k}$ such that the probability of the attribute value in the cluster, $P(A_i = v_{i,k}|C_x)$, is maximum and close to 100%. Therefore, a cluster can be represented by these attribute values. Consider the following coordinate system where the X coordinate axis represents the attributes, the Y-axis represents attribute values corresponding to respective attributes, and the Z-axis represents the probability that an attribute value is in a cluster. Note that for different attributes, the Y-axis represents different attribute value sets. In this coordinate system, we can denote a cluster by a list of d 3D coordinates, $(i, v_{i,k}, P(v_{i,k}|C_x))$, $i = 1, \ldots, d$, where d denotes the number of attributes in the data set. Connecting these d points, we get a graphic line in 3D. Different clusters can then be displayed in 3D space to observe their closeness.

This method, which presents only attribute values with the highest proportions, simplifies the visualization of a cluster. Through operations like rotation or up/down movement, users can then observe the closeness of s-clusters from various angles and decide whether or not they should be grouped in one cluster. Graphic presentation can convey more information than words can describe. Users can obtain reliable thresholds for clustering since the effects of various thresholds can be directly observed in the interface.

4.2. Building a coordinate system

To display a set of s-clusters in a space, we need to construct a coordinate system such that interference among lines (different s-clusters) can be minimized in order to observe closeness. The procedure is as follows. First, we examine the attribute value with the highest proportion for each cluster. Then, summarize the number of distinct attribute values for each attribute, and then sort them in increasing order. Attributes with the same number of distinct attribute values are further ordered by the lowest value of their proportions. The attributes with the least number of attribute values are arranged in the middle of the X-axis and others are put at two ends according to the order described above. In other words, if the attribute values with the highest proportion for all s-clusters are the same for some attribute A_k, this attribute will be arranged in the middle of the X-axis. The next two attributes are then arranged at the left and right of A_k.

After the locations of attributes on the X-axis are decided, the locations of the corresponding attribute values on the Y-axis are arranged accordingly. For each s-cluster, we examine the attribute value with the highest proportion for each attribute. If the attribute value has not been seen before, it is added to the "presenting list" (initially empty) for that attribute. Each attribute value in the presenting list has a location as its order in the list. That is, not every attribute value has a location on the Y-axis. Only attribute values with the highest proportion for some clusters have corresponding locations on the Y-axis. Finally, we represent a s-cluster C_x by its d coordinates $(L_x(i), L_y(v_{i,k}), P(v_{i,k}|C_x))$ for $i = 1, \ldots, d$, where the function $L_x(i)$ returns the X-coordinate for attribute A_i, and $L_y(v_{i,k})$ returns the Y-coordinate for attribute value $v_{i,k}$.

In Fig. 4 for example, two s-clusters and their attribute distributions are shown in (a). Here, the number of distinct attribute values with the highest proportion is 1 for all attributes except for A_2 and A_7. For these attributes, they are further ordered by their lowest proportions. Therefore, the order for these eight attributes are A_5, A_6, A_8, A_1, A_3, A_4, A_7, A_2. With A_5 as center, A_6 and A_8 are

	A1	A2	A3	A4	A5	A6	A7	A8
s1	a 90%	P 100%	S 40%	x 60%	M 100%	α 100%	G 100%	B 99%
	b 10%		F 30%	q 40%				C 1%
			D 30%					
s2	a 100%	O 80%	S 100%	x 40%	M 100%	α 100%	H 100%	B 100%
		P 20%		q 30%				
				z 30%				

(a)

A2	A4	A1	A6	A5	A8	A3	A7

(b)

s1	(1, 1, 1)	(2, 1, 0.6)	(3, 1, 0.9)	(4, 1, 1.0)	(5, 1, 1.0)	(6, 1, 0.9)	(7, 1, 0.4)	(8, 1, 1.0)
s2	(1, 2, 0.8)	(2, 1, 0.4)	(3, 1, 1.0)	(4, 1, 1.0)	(5, 1, 1.0)	(6, 1, 1.0)	(7, 1, 1.0)	(8, 2, 1.0)

(c)

Fig. 4. Example of constructing a coordinate system: (a) two s-clusters and their distribution table; (b) rearranged *X*-coordinate; (c) 3D-coordinates for s1 and s2.

arranged to the left and right, respectively. The rearranged order of attributes is shown in Fig. 4(b). Finally, we transform cluster s_1, and then s_2 into the coordinate system we build, as shown in Fig. 4(c). Taking A_2 for example, there are two attribute values *P* and *O* to be presented. Therefore, *P* gets a location 1 and *O* a location 2 at *Y*-axis. Similarly, *G* gets a location 1 and *H* a location 2 at *Y*-axis for A_7.

Fig. 5 shows an example of three s-clusters displayed in one window before (a) and after (b) the attribute rearrangement. The thicknesses of lines reflect the size of the s-clusters. Compared to the coordinate system without rearranging attributes, s-clusters are easier to observe in the new coordinate system since common points are located at the center along the *X*-axis presenting a trunk for the displayed s-clusters. For dissimilar s-clusters, there will be a small number of common points, leading to a short trunk. This is an indicator whether the displayed clusters are similar and this concept will be used in the interactive analysis described next.

4.3. Interactive visualization and analysis

The CDCS's interface, as described above, is designed to display the merging result of s-clusters such that users know the effects of adjusting merging parameters. However, instead of showing all s-clusters in the first step, our visualization tool displays only two groups from the merging result. More specifically, our visualization tool presents two groups in two windows for observing. The first window displays the group with the highest number of s-clusters since this group is usually the most complicated case. The second window displays the group which contains the cluster pair with the lowest similarity. The coordinate systems for the two groups are conducted respectively.

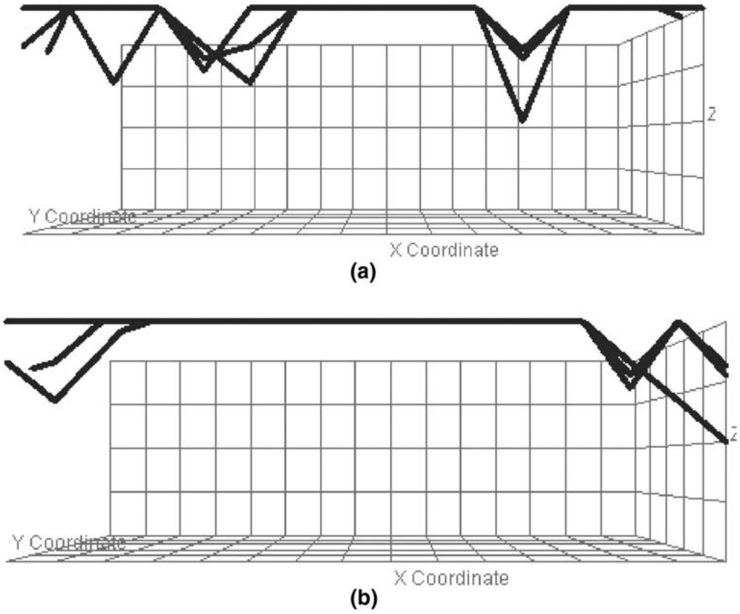
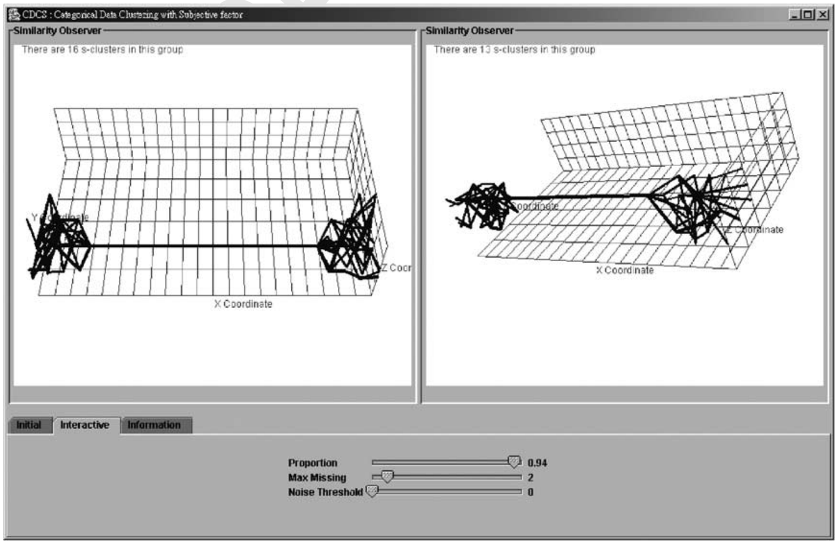


Fig. 5. Three s-clusters: (a) before and (b) after attribute rearrangement.

Fig. 6 shows an example of the CDCS's interface. The data set used is the Mushroom database taken from the UCI machine learning repository [1]. The number of s-clusters obtained from the first step is 106. The left window shows the group with the largest number of s-clusters, while the right window shows the group with the least similar s-cluster pair. The number of s-clusters for


CDCS : Categorical Data Clustering with Subjective factor

Similarity Observer

There are 16 s-clusters in this group

Similarity Observer

There are 13 s-clusters in this group

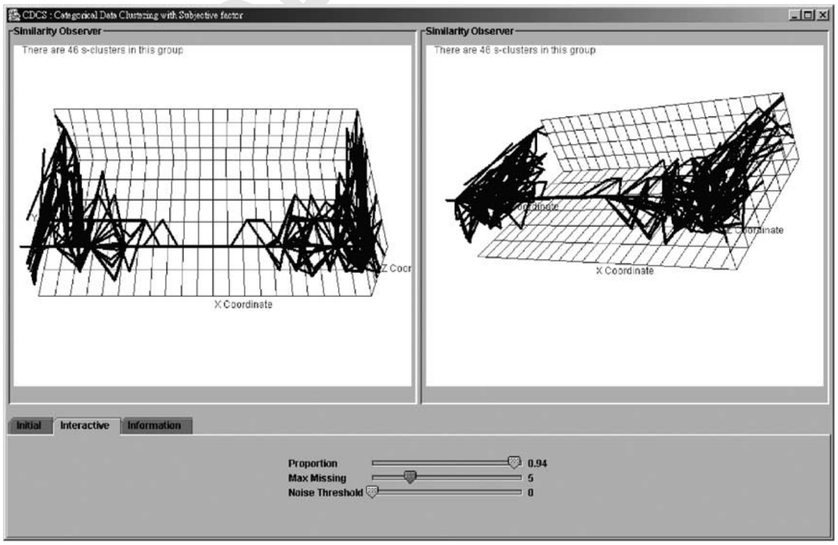
Proportion 0.94
Max Missing 2
Noise Threshold 0

Fig. 6. Visualization of the mushroom data set ($e' = 2$).

these groups are 16 and 13, respectively, as shown at the top of the windows. Below these two windows, three sliders are used to control the merging parameters for group merging and visualization. The first two sliders denote the parameters p' and e' used to control the similarity threshold g'. The third slider is used for noise control in the visualization so that small s-clusters can be omitted to highlight the visualization of larger s-clusters. Each time the slider is moved, the binary matrix BM is recomputed and merging result is updated in the windows. Users can also lock one of the windows for comparison with a different threshold.

A typical process for interactive visualization analysis with CDCS is as follows. We start from a strict threshold g' such that the displayed groups are compact; and then relax the similarity threshold until the displayed groups are too complex and the main trunk gets too short. A compact group usually contains a long trunk such that all s-clusters in the group have the same values and high proportions for these attributes. A complex group, on the other hand, presents a short trunk and contains different values for many attributes. For example, both groups displayed in Fig. 6 have obvious trunks which are composed of sixteen common points (or attribute values). For a total of 22 attributes, 70% of the attributes have the same values and proportions for all s-clusters in the group. Furthermore, the proportions of these attribute values are very high. Through rotation, we also find that the highest proportion of the attributes on both sides of the trunk is similarly low for all s-clusters. This implies that these attributes are not common features for these s-clusters. Therefore, we could say both these groups are very compact since these groups are composed of s-clusters that are very similar.

If we relax the parameter e' from 2 to 5, the largest group and the group with least similar s-clusters refer to the same group which contains 46 s-clusters, as shown in Fig. 7. For this merging threshold, there is no obvious trunk for this group; and some of the highest proportions near the trunk are relatively high, while others are relatively low. In other words, there are no common features for these s-clusters, and thus this merge threshold is too relaxed since different s-clusters



There are 46 s-clusters in this group

There are 46 s-clusters in this group

Proportion 0.94
Max Missing 5
Noise Threshold 0

Fig. 7. Visualization of the mushroom data set with a mild threshold $e' = 5$.

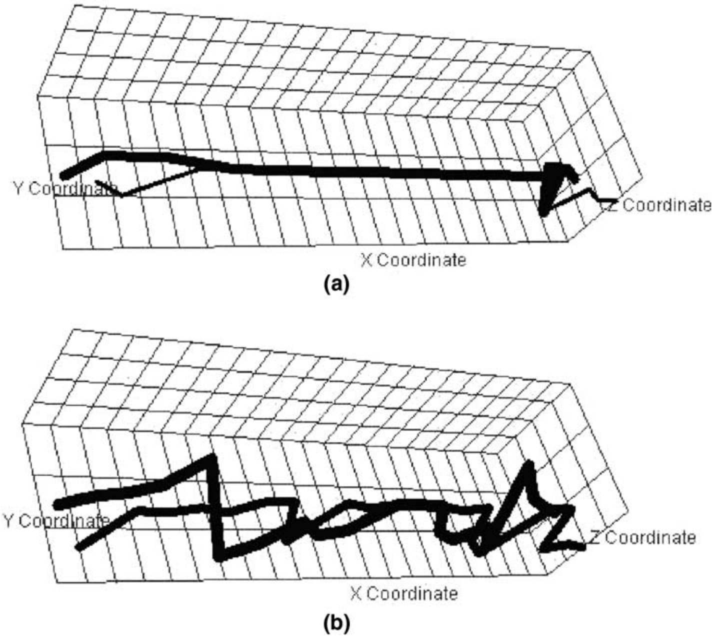


Fig. 8. The most similar group pair for various merging threshold: (a) > (b).

are put in the same group. Therefore, the merging threshold in Fig. 6 is better than the one in Fig. 7.

In summary, whether the s-clusters in a group are similar is based on users' viewpoints on the obvious trunk. As the merging threshold is relaxed, more s-clusters are grouped together and the trunks of both windows get shorter. Sometimes, we may reach a stage where the merge result is the same no matter how the parameters are adjusted. This may be an indicator of a suitable clustering result. However, it depends on how we view these clusters since there may be several such stages. More discussion on this problem is presented in Section 5.2.

Omitting other merged groups does not do any harm since the smaller groups and the more similar groups often have more common attribute values than the largest and the least similar groups. However, to give a global view of the merged result, CDCS also offers a setting to display all groups or a set of groups in a window. Particularly, we present the group pair with the largest similarity as shown in Fig. 8, where (a) presents a high merging threshold and (b) shows a low merging threshold for the data set mushroom. In principle, these two groups must disagree in a certain degree or they might be merged by reducing the merging threshold. Therefore, users decide the right parameter setting by finding the balance point where the displayed complex clusters have long trunk, while the most similar group pair has very short or no trunk at all.

5. Cluster validation

Clustering is a field of research where its potential applications pose their own special requirements. Some typical requirements of clustering include scalability, minimal requirements for domain knowledge to determine input parameters, ability to deal with noisy data, insensitivity to the order of input records, high dimensionality, interpretability and usability, etc. Therefore, it is desirable that CDCS is examined under these requirements.

- First, in terms of scalability, the execution time of the CDCS algorithm is mainly spent on the first step. Simple cluster seeking requires only one database scan. Compared to EM-based algorithm such as AutoClass, which requires hundreds of iterations, this is especially desirable when processing large data sets.
- Second, the interactive visualization tool requires users with low domain knowledge to determine merging parameters.
- Third, the probability-based computation of similarity between objects and clusters can be easily extended to higher dimensions. Meanwhile, the clusters of CDCS can be simply described by the attribute–value pairs of high frequencies, suitable for conceptual interpretation.
- Finally, simple cluster seeking is sensitive to the order of input data, especially for skewed data. One way to alleviate this effect is to set a larger similarity threshold. The effect of parameter setting will be discussed in Section 5.3.

In addition to the requirements discussed above, the basic objective of clustering is to discover significant groups present in a data set. In general, we should search for clusters whose members are close to each other and well separated. The early work on categorical data clustering [9] adopted an *external criterion* which measures the degree of correspondence between the clusters obtained from our clustering algorithms and the classes assigned a priori. The proposed measure, *clustering accuracy*, computes the ratio of correctly clustered instances of a clustering and is defined as

$$\frac{\sum_{i=1}^{k} c_i}{n} \tag{5}$$

where c_i is the largest number of instances with the same class label in cluster i, and n is the total number of instances in the data set.

Clustering accuracy is only an indication of the intra-cluster consensus since high clustering accuracy is easily achieved for larger numbers of clusters. Therefore, we also define two measures using the data's interior criterion. First, we define intra-cluster cohesion for a clustering result as the weighted cohesion of each cluster, where the cohesion for a cluster C_k is the summation of the highest probability for each dimension as shown below:

$$\text{intra} = \frac{\sum \text{in}_k * |C_k|}{n}, \quad \text{in}_k = \frac{\sum_{i=1}^{d} (\max_j P(v_{i,j}|C_k))^3}{d} \tag{6}$$

We also define inter-cluster similarity for a clustering result as the summation of cluster similarity for all cluster pairs, weighted by the cluster size. The similarity between two clusters, C_x and C_y, is as defined in Eq. (3). The exponent $1/d$ is used for normalization since there are d component multiplications when computing $\text{Sim}(C_x, C_y)$.

$$\text{inter} = \frac{\sum_x \sum_y \text{Sim}(C_x, C_y)^{1/d} * |C_x \cup C_y|}{(k-1) * n} \tag{7}$$

We present an experimental evaluation of CDCS on five real-life data sets from the UCI machine learning repository [1]. Four users are involved in the visualization analysis to decide a proper grouping criterion. To study the effect due to the order of input data, each data set is randomly ordered to create four test data sets for CDCS. The result is compared to AutoClass [2] and k-mode [9], where the number of clusters required for k-mode is obtained from the clustering result of CDCS.

5.1. Clustering quality

The five data sets used are Mushroom, Soybean-small, Soybean-large, Zoo and Congress voting, which have been used for other clustering algorithms. The size of the data set, the number of attributes and the number of classes are described in the first column of Table 1. The mushroom data set contains two class labels: poisonous and edible and each instance has 22 attributes. The soybean-small and soybean-large contains 47 and 307 instances, respectively and each instance is described by 35 attributes. (For soybean-small, there are 14 attributes which have only one value, therefore these attributes are removed.) The number of class labels for soybean-small and soybean-large are four and 19, respectively. The zoo data set contains 17 attributes for 101 animals.

Table 1
Number of clusters and clustering accuracy for three algorithms

	# of clusters		Accuracy		
	AutoClass	CDCS	AutoClass	k-Mode	CDCS
Mushroom	22	21	0.9990	0.9326	0.996
22 attributes	18	23	0.9931	0.9475	1.0
8124 data	17	23	0.9763	0.9429	0.996
2 labels	19	22	0.9901	0.9468	0.996
Zoo	7	7	0.9306	0.8634	0.9306
16 attributes	7	8	0.9306	0.8614	0.9306
101 data	7	8	0.9306	0.8644	0.9306
7 labels	7	9	0.9207	0.8832	0.9603
Soybean-small	5	6	1.0	0.9659	0.9787
21 attributes	5	5	1.0	0.9361	0.9787
47 data	4	5	1.0	0.9417	0.9574
4 labels	6	7	1.0	0.9851	1.0
Soybean-large	15	24	0.664	0.6351	0.7500
35 attributes	5	28	0.361	0.6983	0.7480
307 data	5	23	0.3224	0.6716	0.7335
19 labels	5	21	0.3876	0.6433	0.7325
Congress voting	5	24	0.8965	0.9260	0.9858
16 attributes	5	28	0.8942	0.9255	0.9937
435 data	5	26	0.8804	0.9312	0.9860
2 labels	5	26	0.9034	0.9308	0.9364
Average			0.8490	0.8716	0.9260

Table 2
Comparison of AutoClass and CDCS

Data set	Intra		Inter		Intra/inter	
	AutoClass	CDCS	AutoClass	CDCS	AutoClass	CDCS
Mushroom	0.6595	0.6804	0.0352	0.0334	18.7304	*20.3704
Zoo	0.8080	0.8100	0.1896	0.2073	*4.2663	3.9070
Soybean-small	0.6593	0.7140	0.1840	0.1990	3.5831	3.5879
Soybean-large	0.5826	0.7032	0.1667	0.1812	3.4940	*3.8807
Congress voting	0.5466	0.6690	0.1480	0.3001	*3.6932	2.2292

After data cleaning, there are 16 attributes and each data object belonging to one of seven classes. The Congress voting data set contains the votes of 435 congressman on 16 issues. The congressman are labelled as either Republican or Democratic.

Table 1 records the number of clusters and the clustering accuracy for the five data sets. As shown in the last row, CDCS has better clustering accuracy than the other two algorithms. Furthermore, CDCS is better than k-mode in each experiment given the same number of clusters. Compared with AutoClass, CDCS has even more clustering accuracy since it finds more clusters than AutoClass, especially for the last two data sets. The main reason for this phenomenon is that CDCS reflects the user's view on the degree of intra-cluster cohesion. Various clustering results, say nine clusters and 10 clusters, are not easily observed in this visualization method. Therefore, if we look into the clusters generated by these two algorithms, CDCS has better intra-cluster cohesion for all data sets; whereas AutoClass has better cluster separation (smaller inter-cluster similarity) on the whole, as shown in Table 2. In terms of intra-cluster similarity over inter-cluster similarity, AutoClass performs better on Zoo and Congress voting, whereas CDCS performs better on Mushroom and Soybean-large.

5.2. Discussion on cluster numbers

To analyze the data sets further, we record the number of clusters for each merging threshold of the second step. The merging thresholds, g', are calculated by Eq. (4), where p' and e' vary from 0 to 0.99 and 1 to 4, respectively. A total of 400 merging thresholds are sorted in decreasing order. The number of clusters for each merging threshold is recorded until the number clusters reaches three. The way the merging thresholds are calculated avoids steep curves where the number of clusters changes rapidly with small merging thresholds. As shown in Fig. 9(a), we can see five smooth curves with steep downward slopes at the zero ends. The small merging thresholds are a result of the similarity function between two s-clusters (Eq. (3)) where a series multiplications are involved (each factor represents the percentage of common values of an attribute). Therefore, we also change the scale of the merging threshold g' to $\sqrt[d]{g'}$ (d is the number of dimensions), which represents the average similar of an attribute as shown in Fig. 9(b).

We try to seek smooth fragments for each curve where the number of clusters does not change with the varying merging thresholds. Intuitively, these smooth levels may correspond to some macroscopic views where the number of clusters are persuasive. For example, the curve of Zoo has a smooth level when the number of clusters are 9, 8, 7, etc., Mushroom has a smooth level

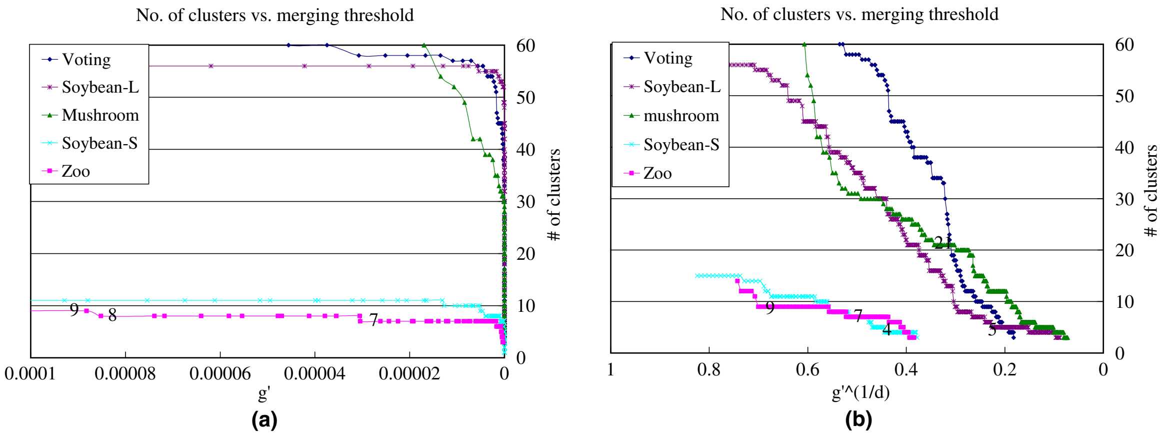


Fig. 9. Number of clusters vs. merging threshold.

at 30, 21, 20, and 12, while the curve of Soybean-small has a smooth level at 7, 5, 4, etc. Some of these coincide with the clustering results of AutoClass and CDCS. Note that the longest fragment does not necessarily symbolize the best clustering result since it depends on how we compute the similarity and the scale of the *X*-axis.

We find that Soybean-large has a smooth fragment when the number of cluster is 5 which corresponds to that of AutoClass, however the average similarity of an attribute is dropped below 0.22. We also notice that the Congress voting has quite steep slope at cluster number between 30 and 5. These may indicate that the data set itself contains a complex data distribution such that no obvious cluster structure is present. We believe that the optimal cluster number varies with different clustering criteria, a decision for the users. For Congress voting, high clustering accuracy is more easily achieved since the number of class labels is only two. As for Soybean-large, clustering accuracy cannot be high if the number of clusters is less than 19. Note that class labels are given by individuals who categorize data based on background domain knowledge. Therefore, some attributes may weight more heavily than others. However, most computations of the similarity between two objects or clusters gives equal weight to each attribute. We intend to extend our approach to investigate these further in the future.

5.3. Parameter setting for dynamic clustering

In this section, we report experiments on the first step to study the effect of input order and skewed cluster size versus various threshold setting. For each data set, we prepare 100 random orders of the same data set and run the dynamic clustering algorithms 100 times for $p = 0.9$, $p = 0.8$, $p = 0.7$, respectively. The mean number of clusters as well as the standard deviation are shown in Table 3. The number of clusters generated increases as the merging threshold p increases. For mushroom, the mean number of clusters varies from 49 to 857 when p varies from 0.7 to 0.9.

To study the effect of skewed cluster sizes, we conduct the following experiment: we run the simple clustering twice with the data order reversed for the second run. To see if the clustering is more or less the same, we use a confusion table V with as rows the clusters in the first run, and as col-

Table 3
The mean number of clusters and its standard deviation for various p

# of clusters	$p = 0.7$		$p = 0.8$		$p = 0.9$	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
Mushroom	49.09	2.29	165.75	4.16	857.98	6.75
Zoo	9.34	1.32	12.96	2.06	16.28	2.47
Soybean-small	16.14	1.67	21.66	2.46	26.296	2.87
Soybean-large	77.64	2.29	108.71	5.66	151.26	8.11
Congress voting	69.62	2.33	97.27	5.33	130.90	7.30

umns the clusters in the second run. Then, the entry i, j corresponds to the number of data objects that were in cluster i in the first experiment, and in cluster j in the second experiment.

The consistency of the two runs can be measured by the percentage of zero entries in the confusion table. However, since two large numbers of small clusters tend to have large zero percentage compared to two small numbers of large clusters, these values are further normalized by the largest number of possible values. For example, consider an $m \times n$ confusion table, where m and n denotes the number of clusters generated for the two runs of the reverse order experiment. The largest number of zero entries will be $(m * n - \max\{m, n\})$.

Let V denotes the confusion table for the two runs. Therefore, the normalized zero percentage (NZP) of the confusion table is defined by

$$\mathrm{NZP}(V) = \frac{|\{(i,j)|V(i,j) = 0, 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n\}|}{m * n - \max\{m, n\}} \tag{8}$$

We show the mean NZP and its standard deviation of 100 reverse order experiments for $p = 0.7$ and $p = 0.9$ with the five data sets in Table 4. For each data set, we also prepare a skewed input order where the data objects of the largest class are arranged one by one, then come the data objects of the second largest class, and so on. The NZP and the numbers of clusters for the reverse

Table 4
Comparison of NZP for skew input order and random order

NZP	Rand vs. Rand		Skew vs. Skew		Rand vs. Skew	
	Mean	S.D.	NZP	$m \times n$	Mean	S.D.
(a) p = 0.7						
Mushroom	0.9904	0.0372	0.9962	44 × 25	0.9894	0.0244
Zoo	0.9036	0.1676	0.9166	10 × 7	0.9100	0.1229
Soybean-small	0.9658	0.0927	0.9553	16 × 15	0.9665	0.0917
Soybean-large	0.9954	0.0282	0.9949	85 × 66	0.9945	0.0244
Congress voting	0.9887	0.0398	0.9878	69 × 50	0.9904	0.0281
(b) p = 0.9						
Mushroom	0.9986	0.0071	0.9980	489 × 423	0.9986	0.0100
Zoo	0.9857	0.0695	0.9893	26 × 19	0.9869	0.0620
Soybean-small	0.9944	0.0360	0.9956	33 × 29	0.9963	0.0373
Soybean-large	0.9993	0.0283	0.9994	233 × 225	0.9994	0.0077
Congress voting	0.9986	0.0100	0.9986	196 × 161	0.9987	0.0100

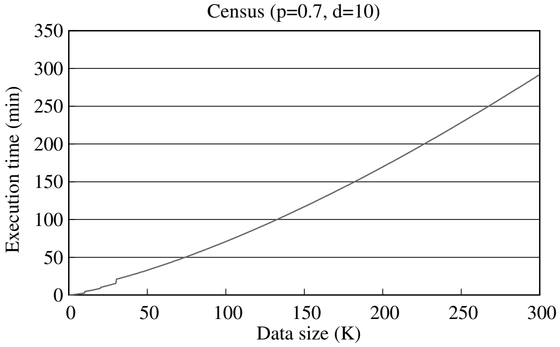


Fig. 10. The execution time required for simple cluster seeking.

order experiments of the skewed input order are displayed in the middle columns (Skew vs. Skew column) of the table for comparison with the average cases (Rand vs. Rand column) for each data set. We also show the mean NZP and its variance between the skew input order and each random order (Rand vs. Skew column). From the statistics, there is no big difference in the skewed input order and average cases. Comparing various *p*: (a) 0.7 and (b) 0.9, the mean NZP increases as the parameter *p* increases. This validates our claim in Section 3 that the effect of input order decreases as the threshold increases.

Finally, we use the Census data set in the UCI KDD repository for scalability experiment. This data set contains weighted census data extracted from the 1994 and 1995 current population surveys conducted by the US Census Bureau. There are a total of 199,523 + 99,762 instances, each with 41 attributes. Fig. 10 shows the execution time for simple cluster seeking of the Census data set with increasing data size. It requires a total of 300 minutes to cluster 299,402 objects with parameter setting $p = 0.7$ and $e = 10$. Note that CDCS is implemented using Java and no code optimization has been used. The clustering accuracy for the first step is 0.9406. For the cluster merging step, it costs 260 seconds to merge 5865 s-clusters for the first visualization, and 64 seconds for the second visualization. After users' subjective judgement, a total of 359 clusters are generated. Comparing to AutoClass, it takes more than two days before it completes the clustering.

6. Conclusion and future work

In this paper, we introduced a novel approach for clustering categorical data with visualization support. First, a probability-based concept is incorporated in the computation of object similarity to clusters; and second, a visualization method is devised for presenting categorical data in a 3D space. Through an interactive visualization interface, users can easily decide a proper parameter setting. Thus, human subjective adjustment can be incorporated in the clustering process. From the experiments, we conclude that CDCS performs quite well compared to state-of-the-art clustering algorithms. Meanwhile, CDCS successfully handles data sets with significant differences in the sizes of clusters such as Mushroom. In addition, the adoption of naive-Bayes classification makes CDCS's clustering results much more easily interpreted for conceptual clustering.

This visualization mechanism may be adopted for other clustering algorithms which require parameter adjustment. For example, if the first step is replaced by complete-link hierarchical clustering with high similarity threshold, we will be able to apply the second step and the visualization technique to display the clustering result and let users decide a proper parameter setting. Another feature that can be included in CDCS is the figure of some clustering validation measure versus our merging threshold. Such measures will enhance users' confidence on the clustering result. In the future, we intend to devise another method to enhance the visualization of different clusters. Also, we will improve the CDCS algorithm to handle data with both categorical and numeric attributes.

Acknowledgement

This paper was sponsored by National Science Council, Taiwan under grant NSC92-2524-S-008-002.

References

[1] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, <http://www.cs.uci.edu/~mlearn/MLRepository.html>, Department of Information and Computer Science, University of California, Irvine, CA, 1998.

[2] P. Cheeseman, J. Stutz, Bayesian classification (autoclass): theory and results, in: Proceedings of Advances in Knowledge Discovery and Data Mining, 1996, pp. 153–180.

[3] D. Fisher, Improving inference through conceptual clustering, in: Proceedings of AAAI-87 Sixth National Conference on Artificial Intelligence, 1987, pp. 461–465.

[4] M. Friendly. Visualizing categorical data: data, stories, and pictures, in: SAS Users Group International, 25th Annual Conference, 2002.

[5] V. Ganti, J. Gehrke, R. Ramakrishnan, Cactus—clustering categorical data using summaries, in: Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 73–83.

[6] D. Gibson, J. Kleinberg, P. Raghavan, Clustering categorical data: an approach based on dynamical systems, VLDB Journal 8 (1998) 222–236.

[7] S. Guha, R. Rastogi, K. Shim, Rock: a robust clustering algorithm for categorical attributes, Information Systems 25 (2000) 345–366.

[8] E.-H. Han, G. Karypis, V. Kumar, B. Mobasher, Clustering based on association rule hypergraphs, in: Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD), 1997, pp. 343–348.

[9] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data Mining and Knowledge Discovery 2 (1998) 283–304.

[10] T. Kohonen, Self-organizing Maps, Springer-Verlag, 1995.

[11] T. Kohonen, S. Kaski, K. Lagus, T. Honkela, Very large two-level SOM for the browsing of newsgroups, in: Proceedings of International Conference on Artificial Neural Networks (ICANN), 1996, pp. 269–274.

[12] A. Konig, Interactive visualization and analysis of hierarchical neural projections for data mining, IEEE Transactions on Neural Networks 11 (3) (2000) 615–624.

[13] W.A. Kosters, E. Marchiori, A.A.J. Oerlemans, Mining clusters with association rules, in: Proceedings of Advances in Intelligent Data Analysis, 1999, pp. 39–50.

[14] S. Ma, J.L. Hellstein, Ordering categorical data to improve visualization, in: IEEE Symposium on Information Visualization, 1999.

[15] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297, 1967.

[16] P.C. Mahalanobis, Proceedings of the National Institute of Science of India 2 (49) (1936).

[17] T.M. Mitchell, Machine Learning, McGraw-Hill, 1997.

[18] C.J. van Rijsbergen, Information Retrieval, Butterworths, London, 1979 (Chapter 3).

[19] E. Sirin, F. Yaman, Visualizing dynamic hierarchies in treemaps. <http://www.cs.umd.edu/class/spring2002/cmsc838f/Project/DynamicTreemap.pdf>, 2002.

[20] J.T. To, R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley Publishing Company, 1974.

[21] A.K.H. Tung, J. Hou, J. Han, Spatial clustering in the presence of obstacles, in: Proceedings of 2001 International Conference on Data Engineering, 2001, pp. 359–367.

[22] K. Wang, C. Xu, B. Liu, Clustering transactions using large items, in: Proceedings of the ACM CIKM International Conference on Information and Knowledge Management, 1999, pp. 483–490.

[23] Y. Zhang, A. Fu, C.H. Cai, P. Heng, Clustering categorical data, in: Proceedings of 16th IEEE International Conference on Data Engineering, 2000, p. 305.

Chia-Hui Chang is an assistant professor at the Department of Computer Science and Information Engineering, National Central University in Taiwan. She received her B.S. in Computer Science and Information Engineering from National Taiwan University, Taiwan in 1993 and got her Ph.D. in the same department in January 1999. Her research interests include information extraction, data mining, machine learning, and Web related research. Her URL is http://www.csie.ncu.edu.tw/~chia/.

Zhi-Kai Ding received the B.S. in Computer Science and Informantion Engineering from National Dong-Hwa University, Taiwan in 2001, and M.S. in Computer Science and Informantion Engineering from National Central University, Taiwan in 2003. Currently, he is working as a software engineer at Hyweb Technology Co., Ltd. in Taiwan. His research interest includes data mining, information retrieval and extraction.