Clustering Categorical Data by Utilizing the Correlated-Force Ensemble

Kun-Ta Chuang and Ming-Syan Chen
Graduate Institute of Communication Engineering
National Taiwan University
Taipei, Taiwan, ROC
E-mail: mschen@cc.ee.ntu.edu.tw, doug@arbor.ee.ntu.edu.tw

Abstract

We explore in this paper a novel clustering algorithm, named CORE (standing for **COR**related-Force **E**nsemble), for categorical data. In general, it is more difficult to perform clustering on categorical data than on numerical data due to the absence of the ordered property in the former. Though several clustering algorithms which concentrate on categorical date were proposed, acquiring the desirable quality remains a challenging issue. Note that there is significance hidden in the correlation between attribute values that can be explored to aid clustering, especially extracting clusters in the high dimensional data. Therefore by employing the concept of correlated-force ensemble, clusters which consist of the highly correlated set of nominal attribute values, can be acquired by the proposed algorithm, CORE. As validated by variant real datasets, it is shown in our experimental results that algorithm CORE significantly outperforms the prior works.

1 Introduction

Mining on databases has attracted a growing amount of research attention due to its wide applicability to improve marketing strategies, business management, and user profile analysis, to name a few [6]. Among others, data clustering is a well-known capacity studied in many research communities, e.g., statistical pattern recognition, machine learning [8], information retrieval [5], and data mining [16]. In essence, clustering aims to retrieve the innate groups formed by similar features of cluster members. Unfortunately, finding optimal clustering result is known to be an NP-hard problem. Thus clustering algorithms usually employ some heuristic processes to find local optimal results. Most clustering techniques utilize a pairwise similarity for measuring the distance of two data points. The widely employed examples are the distance metric, e.g., the Euclidean distance and the Minkowski distance [16]. By utilizing the inherent metric property within numerical data, some clustering algorithms perform very well to find sensible geometric partitions [13][16]. Recently, researches have been elaborated upon the clustering approaches for categorical data, where categorical data are those whose attribute values are nominal and unordered, e.g., color and human hobby. It is noted that difficulties arise in clustering categorical data due to the absence of inherently ordered property of categorical data. Most clustering techniques based on the metric distance measure are thus not applicable to acquire clusters from categorical data domain.

In view of this, algorithms KModes [15], STIRR [10], ROCK [14], CACTUS [9] and COOLCAT [2] have been proposed for clustering categorical data. However, there are many phenomena existing in the reality yet to be explored, e.g., the curse of high dimensionality [3], noisy reality [17], and the disproportionate correlation between different attributes. Consider the weather example shown in Table 1. There are three categories, *Outlook*, *Temperature* and *Humidity*, which have distinct value sets {*Sunny*, *Rainy*}, {*Mild*, *Cool*} and {*Normal*, *High*, *Heigh*}, respectively. From Table 1, we note the *Humidity* value of *6-th* tuple (i.e., r_6) is deemed *noisy* because *High* is wrongly recorded as *Heigh*. A *noisy* data could also be a *missing* value or an *outlier*. Usually, a data cleaning pre-processing work is utilized to intelligently clean the noisy data [17]. However a clustering algorithm with the ability to alleviate the interference from noisy values is highly desirable since the data cleaning work is in general expensive. In addition, attributes *Outlook* and *Humidity* are correlative, e.g., the *Sunny* outlook usually comes with the *Normal* humidity, and the *Rainy* outlook usually comes with the *High* humidity. Such relationship among attributes was not fully explored in most of the previous works, where all attributes tend to be considered independently for clustering.

T_{ID}	Outlook	Temperature	Humidity
r_1	Sunny	Mild	Normal
r_2	Sunny	Mild	Normal
r_3	Sunny	Cool	Normal
r_4	Rainy	Cool	High
r_5	Rainy	Mild	High
r_6	Rainy	Mild	Heigh

Table 1: An illustrative example which includes 6 weather records.

In the following, we briefly describe previous works on clustering categorical data and comment on the problems they might suffer due to the neglect of the above mentioned phenomena. Algorithm KModes is designed as an extension of K-Means algorithm to categorical data, and thus it has the merit of being efficient and the drawback of inadequate quality as similar to K-Means algorithm. In addition, algorithm ROCK is the one of most outstanding clustering algorithms for categorical data nowadays. The concept of *links* is employed in ROCK, and a *link* is defined as the number of common "*neighbors*" between two tuples. Here two tuples are said to be the *neighbor* if their Jaccard-coefficient [16] is larger than or equal to the user defined threshold θ. By utilizing the concept of *links*, algorithm ROCK performs a hierarchical-based clustering procedure to maximize the number of links within clusters and minimize the number of links across different clusters. A similarity criterion is thus conducted under the assumption that data is uniformly distributed. Algorithm ROCK with a good choice of θ usually results in the good quality. However, the threshold θ is difficult to be determined by users [2]. Hence the requirement of a prior knowledge of the data distribution for making appropriate θ needs further justification in real cases. Moreover, algorithm ROCK suffers from the problem which arises when data contains typos, and does not exploit the feature that attributes are correlated because every attribute is viewed as independent to each other.

On the other hand, algorithms STIRR [10] and CACTUS [9] consider that clusters are constructed by a set of distinct categorical values, where each distinct categorical value is called a *node* in [10]. The issue how to cluster tuples is considered as a post-processing work in these approaches. In this perspective, the set of nodes within the same cluster should be closely related. Explicitly, algorithm STIRR is an iterative algorithm based on non-linear dynamic systems. In addition, algorithm CACTUS is devised by using a summarization procedure. This algorithm is achieved by assuming that all attributes are independent, and a node pair $< a_i, a_j >$ is considered *strongly connected* if their co-occurring size is larger than or equal to α times of the expected co-occurring size. Hence, a cluster is considered as a high-density node set in which each node pair is *strongly connected*. It is noted that the advantage of algorithm CACTUS is emphasized in the abilities to deal with high dimensional data and to generate subspace clusters. However, such a cluster definition requires every node pair to be *strongly connected* in a cluster. Consequently, a large number of clusters are likely to be generated and the cluster number may not be close to the desired one in the user perspective. Moreover, as pointed out before, the assumption of having independent attributes would not be applicable in practice.

Algorithm COOLCAT [2] and the algorithm conducted in [7] are recently proposed clustering methods which can deal with categorical data. Both of them achieve clustering on categorical domain by employing the entropy analysis. Explicitly, algorithm COOLCAT is devised as a two-step algorithm, in which a sampled data S are selected to generate cluster seeds in the first step and then the remaining data are incrementally allocated to a suitable cluster in the second step. This incremental procedure will be appropriate to deal with streaming-like data. It is noted that algorithm COOLCAT is devised under the attribute independence assumption, which, however, may need further justification in practice. In addition, high dimension issue and noisy values are not discussed in the paper.

In view of above observations, we make some remarks: (1) Though subspace clusters are more charming in practice [9], they are difficult to be acquired by calculating the similarity of tuples in the original tuple space, e.g., the model used in algorithms ROCK and KModes. A suitable solution is to transform tuples into a set of *nodes*, e.g., the basis model used in algorithm CACTUS. And further, the correlation between those nodes can be analyzed to aid extracting the subspace clustering relationship. (2) By the analysis of the correlation between nodes, the influence of noisy values will be limited because those noisy values are usually distributed randomly and thus have a very small correlation with other values. (3) The assumption that attributes are independent is not required if clusters can be extracted from the correlation between nodes rather than from the relationship between attributes.

With these conclusions, the clustering model by transforming categorical tuples to the described *node* perspective is hence utilized in this paper. Specifically, each node is deemed as a physical object and their degree of correlation is viewed as the magnitude of gravity. The described node perspective is called the *gravity space* in this paper. An abstract illustration of this notion is shown in Figure 1, in which the example in

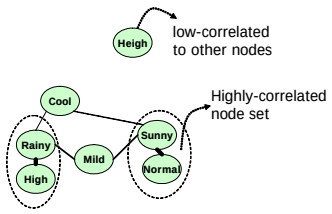

Figure 1: An illustration of transforming tuples to the gravity space.

Parameter	Description
n_a	The distinct node with index a in the dataset
R_{n_a}	The tuple set of node n_a
R_{C_i}	The tuple set in cluster C_i
$Sup(n_a, C_i)$	The support count of n_a in cluster C_i
N_{C_i}	The distinct node set of cluster C_i
$\|N_{C_i}\|$	The number of distinct nodes in cluster C_i
$\|R\|$	Number of tuples in the dataset
$\|C_i\|$	Number of tuples in cluster C_i
$\|N\|$	Number of distinct nodes in the dataset

Table 2: The symbols defined in the paper.

Table 1 is transformed to the *gravity space*. In Figure 1, nodes agglomerate to each other according to the degree of their correlation. Specifically, highly-correlated nodes are close to each other, e.g., nodes *Sunny* and *Normal* are highly-correlated and thus agglomerate together. In contrast, low-correlated nodes would be far from each other, e.g., node *Heigh* is far from other nodes because the noisy node is low-correlated with other nodes.

A gravity-based clustering procedure is thus devised according to the described transformation. Specifically, we classify the correlation of each node pair into two types, namely, *gravitative correlation* and *repulsive correlation*. As such, a cluster similarity measurement is defined as the ensemble of *correlated-forces* between the corresponding two clusters C_i and C_j. Here the *correlated-force* between two nodes in two different clusters is calculated according to their *gravitative/repulsive correlation* and sizes of the two nodes in their corresponding clusters. In brief, the higher the ensemble of *correlated-forces* two clusters have, the more similar to each other they are. This definition of similarity is properly employed in the *gravity space*. A gravity based clustering method, called algorithm *CORE* (standing for **COR**related-Force **E**nsemble) in this paper, is devised from this criterion to perform subspace clustering on categorical tuples. Experimental results show that algorithm CORE outperforms previous algorithms, specially in the presence of various data distributions in real datasets.

2 Preliminaries

2.1 Problem description To begin with, the defined symbols are summarized in Table 2. In this paper, the categorical data is initially represented by a set of tuples. A dataset of tuples is denoted by $R = \{r_1, r_2, ..., r_n\}$, where each tuple r_i consists of a set of distinct nodes from the attribute set $\{A_1, A_2, ..., A_k\}$. Note that attribute A_i consists of a distinct node set $\{a_{i_1}, a_{i_2}, ..., a_{i_j}\}$ with j distinct categorical values. Here we do not differentiate whether two nodes are from the same attribute or not, and treat every node equally. Thus each node is denoted by n_m, with only one index, m, where m is between 1 and $\sum_{1 \le i \le k} |A_i|$. Here $|A_i|$ is the number of distinct categorical values of attribute A_i. Consider the example shown in Table 1, in all there are 7 nodes, {*Sunny, Rainy*} of attribute *Outlook*, {*Mild, Cool*} of attribute *Temperature*, {*Normal, High, Heigh*} of attribute *Humidity*, respectively. Hence the integral index m will be limited in [1,7]. Note that we can acquire the tuple set, R_{n_i}, in which each tuple contains node n_i, after one database scan. For example, node *Sunny* occurs in r_1, r_2, and r_3 in Table 1, and thus R_{Sunny} is $\{r_1, r_2, r_3\}$. In addition, note that a cluster C_i consists of a set of tuples R_{C_i}, and hence we can transform R_{C_i} into the node perspective and get a set of nodes, N_{C_i}:

$$\begin{aligned} N_{C_i} &= \{\forall n_a | n_a \text{ occurs in cluster } C_i\}, \text{ and} \\ |N_{C_i}| &= \text{the size of distinct nodes in cluster } C_i. \end{aligned}$$

Here we call this concept as "*Nodeize*" in this paper to convey the concept that categorical tuples are transformed to the *gravity space*. We will illustrate the detail in Section 2.3.

In the following, we present several preliminary definitions to facilitate our discussions.

Definition 1 (The correlation between nodes n_a and n_b):

The correlation between nodes n_a and n_b is defined as:

$$Corr(n_a, n_b) = \frac{|R_{n_a} \cap R_{n_b}|}{|R_{n_a} \cup R_{n_b}|},$$

where $|R_{n_a} \cap R_{n_b}|$ is the count of tuples which contains both node n_a and n_b, and $|R_{n_a} \cup R_{n_b}|$ is the count of tuples which contains either node n_a or node n_b in R.

For example, the correlation of nodes *Sunny* (occurs in tuples r_1, r_2 and r_3) and *Cool* (occurs in tuples r_3 and r_4) is $\frac{|\{(r1,r2,r3) \cap (r3,r4)\}|}{|\{(r1,r2,r3) \cup (r3,r4)\}|} = \frac{1}{4} = 0.25$. Note that this is the *Jaccard-coefficient measurement* [16], and is able to indicate the correlated level between two nodes. In general, there are many measurements to

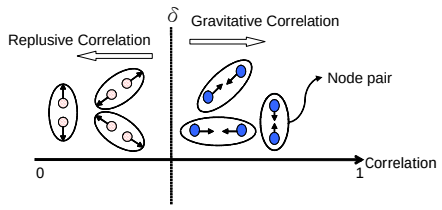

Figure 2: The illustration of the force correlation.

evaluate the correlation between two patterns [19], e.g., the *chi-square* measure. We select Jaccard-coefficient measurement because it satisfies many mathematical properties. Explicitly, the range of correlation will be in 0˜1. The correlation of $n_a \rightarrow n_b$ is equal to the correlation of $n_b \rightarrow n_a$ (*symmetric distance*) because the symmetry property holds. Tuples without nodes n_b and n_a will not be taken in consideration when the correlation of these two nodes is calculated (*null invariance* property). For example, tuples r_5 and r_6 in Table 1 would not be considered into the calculation of the correlation between nodes *Sunny* and *Cool*. In practice, this property allows our mechanism to be extensible to the transactional data model.

Note that $Corr(n_a, n_b)$ is bounded in 0˜1, it is possible to determine a threshold, δ, to divide the values into two intervals, which are the *gravitative correlation* interval and the *repulsive correlation* interval, respectively. Both *gravitative correlation* and *repulsive correlation* are referred to as the *force correlation*.

Definition 2 (The Force Correlation):

The *force correlation* is defined as:

$$FC(n_a, n_b) = Corr(n_a, n_b) - \delta.$$

In other words, if $Corr(n_a, n_b) \geqq \delta$, nodes n_a and n_b are *gravitatively correlated*. Oppositely, if $Corr(n_a, n_b) < \delta$, nodes n_a and n_b will be *repulsively correlated*. With this definition, node pairs will have similar and dissimilar behavior like the definition of similarity and dissimilarity in traditional clustering algorithms. A concept of the force correlation is illustrated in Figure 2, in which the node pairs are divided into two partitions by cutting the correlation axis at δ.

Furthermore, to adequately utilize the behavior of *gravitative* and *repulsive* correlations, we can define the node-gravity formula and node-repulsion formula to represent the gravity/repulsion between each node pair. In essence, the *gravity* concept is exploited to perform clustering in [12][20], which utilized the *gravity* to represent the similarity between two clusters over the numerical domain. In other words, the larger *gravity* between two clusters, the more similar to each other they are. Note that the *gravity* in the physics is generally expressed as $F = \frac{g*m_1*m_2}{d^2}$, where g is a constant, m_1 and m_2 are the *mass* of the corresponding two objects, and d is the *distance* between them. Similarly, we conduct the concept of the *repulsion* to avoid that any two-object pair is always co-attractive because a *gravity* absolutely exists and would merge ultimately.

We utilize the *gravitative correlation* and the *repulsive correlation* to conduct the *gravitative/repulsive* magnitude, or said the *force* magnitude, between two nodes in different clusters. Explicitly, we have the following definition.

Definition 3 (Gravitative/Repulsive force magnitude):

For node n_a in cluster C_i and node n_b in cluster C_j, the *force*, represented by $F = \frac{g*m_1*m_2}{d^2}$, is defined as:

$$F(C_i : n_a, C_j : n_b) = \alpha(n_a, n_b) * \frac{\frac{Sup(n_a, C_i)}{|C_i|} * \frac{Sup(n_b, C_j)}{|C_j|}}{d(n_a, n_b)^2},$$

where $|C_i|$ and $|C_j|$ are the counts of tuples in cluster C_i and C_j, respectively. $Sup(n_a, C_i)$ is the support count of node n_a in cluster C_i, and $Sup(n_b, C_j)$ is the support count of node n_b in cluster C_j. In addition, $\alpha(n_a, n_b)$ is defined as:

$$\alpha(n_a, n_b) = \begin{cases} 1, & \text{if } n_a \text{ and } n_b \text{ are } \textit{gravitatively} \text{ correlated} \\ -1, & \text{if } n_a \text{ and } n_b \text{ are } \textit{repulsively} \text{ correlated} \end{cases},$$

which can indicate that this force between nodes n_a and n_b is *gravity* or *repulsion*.

Note that $\frac{Sup(n_a, C_i)}{|C_i|}$ and $\frac{Sup(n_b, C_j)}{|C_j|}$ are normalized terms and therefore are able to convey the concept of the "*mass*" rather than the "*weight*" in the aspect of physics. Another importance needs to be specified is the reciprocal of distance square, i.e., $\frac{1}{d(n_a, n_b)^2}$. Note that a property of this value must hold: if the correlation of the node pair $< n_a, n_b >$ is larger, the reciprocal of distance square should be larger to represent the relation that nodes n_a and n_b are close to each other.

Definition 4 (The reciprocal of distance square between node n_a and node n_b):

The reciprocal of distance square between node n_a and node n_b is defined as:

$$\frac{1}{d(n_a, n_b)^2} = [Corr(n_a, n_b) - \delta]^2 = FC(n_a, n_b)^2.$$

By adopting this definition, node *gravity/repulsion* defined in Definition 3 would satisfy the property that if $Corr(n_a, n_b)$ is much larger then δ, they would have a strong *gravity* to stand for the behavior that nodes n_a and n_b usually occur together. On the other hand, if $Corr(n_a, n_b)$ is much smaller then δ, they will have a

strong repulsion because nodes n_a and n_b almost do not occur together. Note that $d(n_a, n_b) = d(n_b, n_a)$ because Jaccard-coefficient correlation is *symmetric* [19].

It can be verified that $F(C_i : n_a, C_j : n_b)$ is limited in [-1,1], because $\frac{Sup(n_a,C_i)}{|C_i|} * \frac{Sup(n_b,C_j)}{|C_j|}$ and $\frac{1}{d(I_a,I_b)^2}$ are both bounded in [0,1].

The other question to resolve is to determine the correlation threshold, δ. Thus supported by a statistical perspective and the experimental results, we conduct a recommendable correlation threshold:

Definition 5 (The recommendable correlation threshold):

The recommendable value $\widehat{\delta}$ is conducted as:

$$\widehat{\delta} = \frac{\sum_{n_a \in N} [\sum_{n_b \in N, n_b \neq n_a} Corr(n_a, n_b)]}{|N| * (|N| - 1)},$$

where $|N|$ denotes the size of distinct nodes of the dataset. In brief, $\widehat{\delta}$ is the correlation mean of all node pairs.

Lemma 1 The distribution of correlation tends to be a Gaussian distribution with mean of $\widehat{\delta}$ and variance of σ by following the central limit theorem [18].

Note that according to Lemma 1, the node set would be equally divided into 2 partitions by setting $\delta = \widehat{\delta}$. In our opinion, a sensible δ would satisfy that there are half of node pairs determined *gravitative* and the other half determined *repulsive*. This argument is also strengthened by our experimental observation because setting $\delta = \widehat{\delta}$ usually obtains an excellent quality in different cases.

It is noted that $\widehat{\delta} \pm 2\sigma$ is usually considered *significant* in the statistical sense [18]. Thus another possible suggestion that if the users want to make the data strictly merge together, let $\delta = \widehat{\delta} + \varepsilon$, where ε is a positive real number and smaller than 2σ. On the other hand, let $\widehat{\delta} - 2\sigma < \delta \leq \widehat{\delta}$ if the users want to make the data loosely merge together.

2.2 Clustering objective The objective of clustering categorical data considered in this paper can be stated as "*By a given database of categorical tuples, determining the clustering in such a way that each cluster has the largest intra-gravity force within the cluster and has the largest inter-repulsion force toward other clusters.*" This objective can also be translated to that the cluster C_i should have the largest *gravity* and smallest *repulsion* within the cluster, and also have smallest *gravity* and largest *repulsion* toward other clusters. Clearly, if the node pair $< n_a, n_b >$ has large *gravitative correlation*, n_a and n_b should be arranged to the same cluster because they often occur together in the original dataset. In essence, clusters are able to be

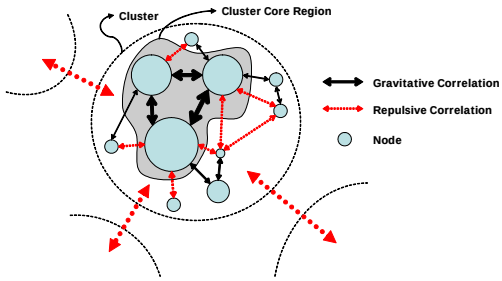

Figure 3: The galaxy representation of clusters.

represented as a *galaxy* under the clustering objective. Figure 3 shows this concept clearly. In Figure 3, some "large" nodes denote those nodes have big *mass* (i.e., the support value of node n_a in cluster C_i, $\frac{Sup(n_a,C_i)}{|C_i|}$, is large), and thicker lines denote the node pairs have the larger *force correlation*. Note that the above illustration method is the same is the same whether the correlation is *gravitative* or *repulsive* in Figure 3. Based on the clustering objective, we know that the cluster is formed by the cluster core region with a node set whose elements have large *gravity* in spite of some nodes within the cluster are repulsive to others. In addition, the force between clusters is expected to be *repulsive* in theory. Note that we omit many lines in Figure 3 which are corresponding to *gravitative* and *repulsive correlation* for ease of presentation. In practice, all node pairs in the cluster will contribute their correlation when performing clustering even some of them are small.

The following definition states the cluster similarity measurement based on the correlated-force ensemble to measure the force between two clusters.

Definition 6 (The Clustering Similarity Measurement based on correlated-force ensemble):

The cluster similarity between clusters C_i and C_j is defined as:

$$L(C_i, C_j) = \frac{\sum_{n_a \in N_{C_i}, n_b \in N_{C_j}} F(C_i : n_a, C_j : n_b)}{|N_{C_i}| * |N_{C_j}|}.$$

$L(C_i, C_j)$ is calculated by summing all forces between clusters C_i and C_j, and normalized by the count of node pairs in these two clusters. The reason why we use the normalized term is to avoid the case that large clusters would be always more similar because their forces are large in general. Note that $F(C_i : n_a, C_j : n_b)$ is limited in [-1,1] that we had mentioned before, $L(C_i, C_j)$ is hence limited in [-1,1]. Explicitly, we consider that the cluster pair $\{C_i, C_j\}$ which has the largest $L(C_i, C_j)$ should be merged first because they are highest co-attractive.

Node1	Node2	Correlation
Sunny	Mid	0.67
Sunny	Cool	0.25
Sunny	Normal	1.00
Sunny	High	0.00
Sunny	Heigh	0.00
Rainy	Mid	0.4
Rainy	Cool	0.25
Rainy	Normal	0.00
Rainy	High	0.67
Rainy	Heigh	0.33
Mild	Normal	0.4
Mild	High	0.2
Mild	Heigh	0.25
Cool	Normal	0.25
Cool	High	0.33
Cool	Heigh	0.00

The Correlation of all other pairs are zero.

(a) Jaccard-coefficient Correlation Table

δ=0.24

Transfer to Force Correlation Table

Node1	Node2	Forced Correlation
Sunny	Mid	0.43
Sunny	Cool	0.01
Sunny	Normal	0.76
Sunny	High	-0.24
Sunny	Heigh	-0.24
Rainy	Mid	0.16
Rainy	Cool	0.01
Rainy	Normal	-0.24
Rainy	High	0.43
Rainy	Heigh	0.09
Mild	Normal	0.16
Mild	High	-0.04
Mild	Heigh	0.01
Cool	Normal	0.01
Cool	High	0.09
Cool	Heigh	-0.24

The forced correlation of all other pairs are -0.24.

(b) Forced Correlation Table

Figure 4: The Force Correlation table.

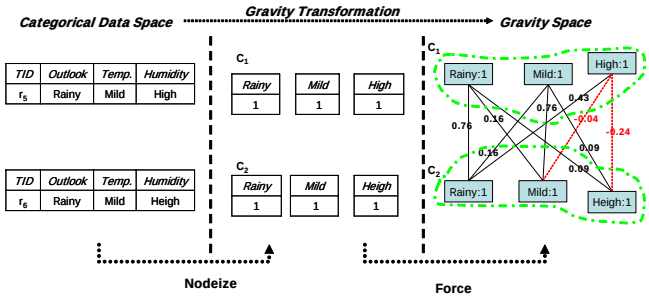

Figure 5: The illustration of Gravity Transformation for clusters consisting of 1 tuple.

2.3 An Illustrative Example We illustrate the concept of the proposed cluster similarity measurement by going through the example shown in Table 1. Initially, the example dataset is scanned once, and thus the Jaccard-coefficient correlations of all node pairs are calculated. The result is shown in the part (a) of Figure 4. In addition, some node pairs never happen, e.g., nodes *Sunny* and *High* do not occur together in the example, and their correlation will thus be zero. Therefore some of those zero-correlation pairs are omitted in Figure 4 to make the illustration sample. Note that Jaccard-coefficient is symmetric, such that the node pair $< n_a, n_b >$ is also omitted in the table if the node pair $< n_b, n_a >$ is shown already. In short, totally there are 7 nodes and $P_2^7 = 7 * 6 = 42$ possible permutations of node pairs. As a result, the recommendable δ defined in Definition 5 is calculated as $\widehat{\delta} = \frac{0.67+0.25+1.00+...+0.00}{7*6} = 0.24$, which is the correlation mean of all node pairs. Therefore we transfer the correlation to the force correlation by decreasing the correlation by 0.24. The result of the *force correlation* table is shown is the part (b) of Figure 4.

After retrieving the information of *force correlations*, we would like to calculate the *similarity* between two clusters (In the hierarchical clustering model, each tuple denotes a cluster initially). The overall process is called as "*Gravity Transformation*" in this paper. Specifically, the cluster members are transformed from the original categorical data space to the *gravity space,* and thus the similarity between clusters are calculated in this procedure. One example of the process is illustrated in Figure 5. Two sub-procedures, i.e., *Nodeize* and *Force*, are included in the *Gravity Transformation*. In the *Nodeize* procedure, tuples are transformed into a set of nodes. Here two tuples r_5 (be labeled by cluster C_1) and r_6 (be labeled by cluster C_2) are considered. They are transformed to the node perspective by the *Nodeize* procedure at first. Theoretically there is only one occurrence of each node in a tuple. In the following, the *force magnitudes* of nodes n_a and n_b are accumulated in the *Force* procedure, where $n_a \ni N_{C_1}$ and $n_b \ni N_{C_2}$. Here the force correlations are labeled in the corresponding line between the node pairs. For example, the node pairs $< C_1 : Rainy, C_2 : Heigh >$, $< C_1 : Rainy, C_2 : Mild >$ has force correlations 0.09 and 0.16, respectively. Thus $F(C_1 : Rainy, C_2 : Heigh) = 1 * 1 * 0.09^2 = 0.0081$ and $F(C_1 : Rainy, C_2 : Mild) = 1 * 1 * 0.16^2 = 0.0256$. Note that $F(C_1 : Rainy, C_2 : Heigh)$ is much smaller than $F(C_1 : Rainy, C_2 : Mild)$, which can demonstrate our clustering criterion would mitigate the influence of noisy values. $L(C_1, C_2)$ can be calculated sequentially as $\frac{0.0256+0.0081+...-0.0016-0.0576}{3*3} = 0.1498$, where there are $3 * 3$ node pairs between those two clusters.

In practice, clusters would consist of more than one tuple. Without loss of generality, we give an example of clusters consisting of 2 tuples in Figure 6, where cluster C_1 consists of tuples r_1, r_2 and cluster C_2 consists of tuples r_5, r_6. Those tuples are transformed to the node perspective by the *Nodeize* procedure and thus get the information of nodes, e.g., node *Sunny* occurs twice in cluster C_1, i.e., $Sup(Sunny, C_1) = 2$, and node *High* occurs only once in cluster C_2, i.e., $Sup(High, C_2) = 1$. In the following, $F(C_1 : n_a, C_2 : n_b)$ of node n_a in C_1 and node n_b in C_2 is calculated by injecting the support of nodes, i.e., the concept of *mass* said in the definition 3. For example, $F(C_1 : Sunny, C_2 : High) = -1 * \frac{2}{2} * \frac{1}{2} * (-0.24)^2 = -0.0288$. It is a negative value and contributes a *repulsion* because these two nodes are never occurring together in Table 1. From the *Force* procedure shown in the right part of Figure 6, we know that only the node *Mild* would have the *gravity* and other nodes all contribute *repulsion*. Therefore the cluster pair $\{C_1, C_2\}$ is not considered a good candidate to merge.

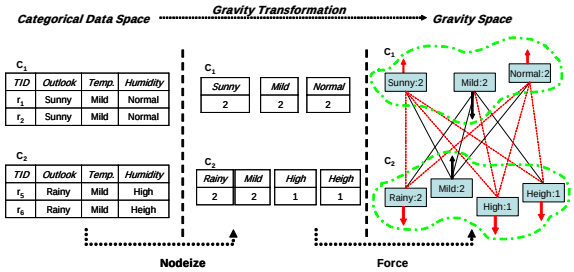

Figure 6: The illustration of Gravity Transformation for clusters consisting of 2 tuples.

3 Design of Algorithm CORE

In light of the clustering objective, we develop an algorithm, named CORE (standing for clustering by **COR**related-force **E**nsemble), to perform subspace clustering on categorical data. Algorithm CORE is an agglomerative clustering method which transforms categorical tuples into the gravity space and calculates the ensemble of correlated-forces between each cluster pair. The detail of each step is described as follows:

Procedure of Algorithm CORE

Step 1. Scan the database R and generate FCarray.

Explicitly, the element of row a and column b of the *force correlation* table (abbreviated as *FCarray*) denotes the *force correlation* of the node pair $< n_a, n_b >$. *FCarray* will facilitate the computation to obtain the *force magnitude*.

Step 2. Build initial cluster set from R and construct the local heap of each cluster.

Note that algorithm CORE is devised based on the hierarchical clustering model. Therefore initially each tuple in R will construct a cluster, respectively. Theoretically, the local heap of the cluster C_i should contain the similarity between C_i and any other cluster C_j, i.e., $L(C_i, C_j)$. However, the local heap of cluster C_i can merely record the information of those clusters which $L(C_i, C_j) > 0$ in practice. It will be an appropriate space-saving strategy because the users may merge those clusters with gravity in the first clustering run.

Step 3. Rebuild the global heap and make sure it consists of all maximum elements in the local heap of each cluster.

The global heap is constructed by $Max(C_i.LocalHeap)$, where C_i is any remaining cluster and $C_i.LocalHeap$ is the local heap of C_i. $Max(C_i.LocalHeap)$ stands for the cluster pair $\{C_i, C_j\}$ whose $L(C_i, C_j)$ is maximal among all cluster pairs in the local heap of C_i.

Step 4. If the global heap is empty, goto Step 8.

The global heap will be empty if no any cluster pair is considered to merge. In other words, their is no any element in the local heaps of remaining clusters.

Step 5. Extract the maximum cluster pair $\{C_a, C_b\}$ from the head of the global heap. Merge cluster C_b into cluster C_a.

Note that cluster C_a will add those necessary information contained in cluster C_b.

Step 6. For each cluster $C_i \ni \{C_a.LocalHeap \cup C_b.LocalHeap\}$, let the *goodness* of the cluster pair $\{C_i, C_a\}$ be accumulated by the *goodness* of the cluster pair $\{C_i, C_b\}$. In addition, the information of the cluster pair $\{C_i, C_b\}$ is removed from $C_i.LocalHeap$.

The value, named *goodness*, of cluster pair $\{C_i, C_j\}$ is stated as follows:

$$g(C_i, C_j) = \sum_{n_a \in N_{C_i}, n_b \in N_{C_j}} \{\alpha(n_a, n_b) * Sup(n_a, C_i) * Sup(n_b, C_j) * [Corr(n_a, n_b) - \delta]^2\}.$$

Recording *goodness* of every cluster pair $\{C_i, C_p\}$ in their corresponding local heaps will make the below equation hold: The similarity of the cluster pair $\{C_i, C_j\}$, i.e., $L(C_i, C_j)$, is equal to

$$\frac{g(C_i, C_p) + g(C_i, C_q)}{|C_i| * |C_j| * |N_{C_i}| * |N_{C_j}|},$$

where cluster C_j is constituted by clusters C_p and C_q in the early merging procedure.

Note that cluster C_b will be merged into C_a, thus information about cluster C_a in the local heap of cluster C_i would be updated and information about cluster C_b in the local heap of cluster C_b should be removed.

Step 7. Delete cluster C_b. Then go back to Step 3.

After updating the information about merging cluster C_b into cluster C_a, C_b would be deleted from the cluster set and go into the next hierarchical merging iteration. It is noted that there is a while loop from Step 3 to Step 7.

Step 8. Perform the outlier detection task.

When the global heap is empty, removing the clusters containing a small number of tuples is one recommendable *outlier detection* strategy. In addition, algorithm CORE can give a recommending number of clusters, which is the result in the end of Step 8. Note that this is an option rather than a necessity because it is possible to generate the user-desired number of clusters by performing Step 9. Note that algorithm ROCK can not merge clusters after getting the recommending result even though the recommending number of clusters is much larger than the user desired one.

Step 9. Rebuild local heaps of all remaining clusters. Then go back to Step 3.

If the recommending number of clusters is not acceptable, algorithm CORE can continue to merge remaining clusters toward the user-desired number. Algorithm CORE rebuilds the local heap of the remaining clusters. Note that all elements of the local heap would merely contain cluster pairs which similarity is smaller than zero because all cluster pairs connected by the *gravity* have been merged in the first clustering run. As a result, merging those cluster pairs which have little repulsion (after going back to Step 3), the desired number of clusters can be generated consequently.

Step 10. Finish the clustering, and report the result.

4 Experimental Results

We assess the quality of algorithms ROCK, KModes and CORE in Windows XP professional platform with 1G memory and 1G P3-CPU. Note that algorithms CACTUS and STIRR did not consider how to allocate data tuples into clusters, and are thus not included into our comparison. Those codes are all implemented by Java and complied by sun jdk1.4. We describe the evaluation model in Section 4.1. The experiments of public domain real data are shown in Section 4.2.

4.1 The Evaluation Model In general, the evaluation model for clustering is based on the *square error criterion* [16]. However, the absence of the natural ordered property of categorical data makes this kind of evaluation models difficult to be employed. Several evaluation models, e.g., *significance test* [16] and *category utility function* [11], are conducted to evaluate clustering on categorical data based on the statistical test. However, the human viewpoint, which is the most important issue should be concerned in practice, is not taken into account in those models. Some previous algorithms conduct experiments by applying the data with a *class* attribute, i.e., the data used in the supervised learning. Similarly, we take the kind of data, and additionally utilize an evaluation function, *F-measure* [1], which is a weighted combination of the recall rate and the precision rate, to conduct the quantifiable analysis. The advantages of this evaluation model include that the factor of the human perspective can be validated and that a quantifiable comparison can be conducted. This evaluation model is stated as follows. Given a set of tuples, each tuple r_i has a class label L_i. There are $\{C_1, C_2, ..., C_p\}$, p distinct class labels and $L_i \subseteq \{C_1, C_2, ..., C_p\}$. After blinding the label L_i and performing the clustering algorithm, we have the clustering result U and an estimated class label $\widehat{L_i}$, i.e., the identification of the allocated cluster, for each tuple r_i. In general, there are $\{E_1, E_2, ..., E_q\}$, q distinct clusters generated and $\widehat{L_i} \subseteq \{E_1, E_2, ..., E_q\}$. Note that $q \geq p$ is permitted. However, creating more clusters than the desired clusters is not desirable. Thus, we select p distinct clusters $\{C_1^*, C_2^*, ..., C_p^*\}$ from $\{E_1, E_2, ..., E_q\}$ for corresponding to $\{C_1, C_2, ..., C_p\}$, respectively, and ignore other remaining clusters. Moreover, each pair of matching clusters $< C_i, C_i^* >$ is given a score by applied *F-measure* [1]:

$$f_\beta(C_i, C_i^*) = \frac{(\beta^2 + 1)PR}{\beta^2 P + R},$$

where P and R are the precision rate and the recall rate, respectively. β is a weighted value to adapt the importance of P and R. We set $\beta = 1$ for all cases to equally treat the precision rate and the recall rate. Thus, the average score of this combination is formulated as:

$$F_\beta = \frac{\sum_{i=1}^{p} f_\beta(C_i, C_i^*)}{p}.$$

To denote the best possible situation of this clustering result, the score of the clustering result U is hence selected as the maximum F_β, which is abbreviated as $Max(F_\beta)$, from $(C_p^q * p!)$ combinations. In essence, good clustering quality is able to result in a high $Max(F_\beta)$ because this clustering result attains both the high precision rate and the high recall rate. The detail of this evaluation method will be illustrated in the result shown in Section 4.2.

In addition, the default number of clusters is set to the corresponding number of class labels, i.e., $\#Classes$, of each data in all experiments. Moreover, the results of algorithms KModes are unstable due to the random selection of the initial cluster centers. Thus we perform ten executions of algorithm KModes for each experiment and report the result with the best $Max(F_\beta)$ from the ten executions in our simulation. To assess algorithm ROCK, since there is no predetermined θ, for fair comparisons, we run 19 executions with different θ which is set to 0.05~0.95 and steps by 0.05. Note that a larger θ will cause many tuples have no neighbors and lead to the generation of too many outliers. For equally comparing the three algorithms, we discard the results of algorithm ROCK whose numbers of outliers are more than 15% of tuples, and select the one with $Max(F_\beta)$ from the remainder. Algorithm CORE is executed with the recommended value $\widehat{\delta}$ (i.e., the mean of all node pairs) and no user-defined procedure is required.

4.2 Experiments with Real Data We conduct comparisons on several public domain data downloaded from UCI machine learning repository [4]. Here we select seven categorical data, including the soybean database, the congressional voting records database (abbreviated as *Voting*), mushrooms database, Teaching Assistant Evaluation database (abbreviated as *TAE*), the zoo database, the wisconsin breast cancer database (abbreviated as *Breast-Cancer*), and the lymphography database. Among others, in Figure 7 we show the result of *Soybean* dataset for demonstrating the detail of the clustering evaluation model. We select the small version of the soybean data, which is a well-known dataset in the machine learning community. There are 35 attributes, 47 tuples, and 4 kinds of diseases (denoted by D1, D2, D3 and D4, respectively) as the class labels in the dataset. In addition, there is no missing value in this dataset. The clustering results of algorithms CORE, KModes and ROCK are shown in the part (a), (b) and (c) of Figure 7, respectively. Here $D1^*, D2^*, D3^*$ and $D4^*$ are the outcome clusters, which form the best combination with $Max(F_\beta)$ for representing the four class labels D1, D2, D3 and D4, respectively. Note that in Figure 7, a 4*4 contingence table is shown to represent the tuple distribution of the clustering result. For example, there are 10 tuples belonging to D2, and in the result of algorithm KModes, there are 8 tuples allocated to $D2^*$ and 2 tuples allocated to $D4^*$. According to the information shown in the contingence table, we can calculate the corresponding precision P and recall R, and *F-measure* is conducted. For example, the outcome cluster $D2^*$ of algorithm KModes has 8 tuples belonging to class label D2 and 1 tuple belonging to D3. Thus $D2^*$ has a total of 9 tuples and the precision $P = \frac{8}{9} = 0.89$ when $D2^*$ is used to represent D2. Similarly, the recall $R = \frac{8}{10} = 0.8$ because there are 10 tuples belonging to D2. Finally, the score of *F-measure* for this cluster pair is $f_\beta(D2, D2^*) = \frac{2*P*R}{(P+R)} = 0.84$. As a result, $Max(F_\beta)$ of each clustering result is obtained and shown in the part (d) of Figure 7. For example, $Max(F_\beta)$ of algorithm KModes is calculated by $\frac{0.21+0.84+0.58+0.56}{4} = 0.55$, which is the average value of the four *F-measure* scores.

From this experiment, we get a surprising result that algorithm CORE generates the outcome clusters which perfectly match the four disease labels when we set $\delta = \widehat{\delta} + 2\sigma = 0.35$, which is the upper bound of the recommendable δ. In this situation, algorithm CORE results in the best possible $Max(F_\beta)$, i.e., $Max(F_\beta) = 1$. On the other hand, algorithms ROCK and KModes merely achieve 0.58 and 0.55, respectively. Note that Soybean dataset is a high dimensional dataset (35 attributes). Hence from the experiment we demonstrate

	D1	D2	D3	D4	P	R	F-measure
D1*	**10**	0	0	0	1	1	1
D2*	0	**10**	0	0	1	1	1
D3*	0	0	**10**	0	1	1	1
D4*	0	0	0	**17**	1	1	1

(a) CORE

	D1	D2	D3	D4	P	R	F-measure
D1*	**2**	0	0	7	0.22	0.20	0.21
D2*	0	**8**	1	0	0.89	0.80	0.84
D3*	6	0	**7**	1	0.50	0.70	0.58
D4*	2	2	2	**9**	0.60	0.53	0.56

(b) KModes

	D1	D2	D3	D4	P	R	F-measure
D1*	**7**	0	0	8	0.47	0.70	0.56
D2*	1	**7**	0	0	0.88	0.70	0.78
D3*	1	3	**4**	0	0.50	0.40	0.44
D4*	1	0	6	**9**	0.56	0.53	0.55

(c) ROCK

Data Name	Soybean
#Classes	4
|R|	47
#Attributes	35
Missing	No

	$Max(F_\beta)$
CORE (δ=0.35)	**1.00**
ROCK (θ=0.25)	**0.58**
KModes	**0.55**

(d) Data Summary & Results

Figure 7: The results of Soybean dataset.

the outperformance of algorithm CORE even confronts the curse of high dimensionality [3] because algorithm CORE can extract cluster features from subspace node set.

The remaining experiments of real datasets are summarized in Table 3. Note that the number of classes $\#Classes$, the number of tuples $|R|$, and the number of attributes $\#Attributes$ vary in each dataset. In addition, some datasets contain missing values while others do not. In the case of outliers, the clustering results with outliers are also annotated by the number of outliers (denoted as $|O|$ in Table 3). From the observation on clustering those variant real datasets, algorithm CORE behaves more stably than the other two algorithms. Note that algorithm CORE mostly outperforms algorithms ROCK and KModes. Though algorithm ROCK is the best algorithm for *Voting* and *Zoo* dataset, the results of algorithm CORE are very close to that of algorithm ROCK in the two cases. Explicitly, in the result of *Voting* dataset, there is a cluster with 3 tuples which is generated by algorithm CORE, and thus we consider them as outliers. However, there are 63 tuples determined as outliers by algorithm ROCK. It is a admirable result of algorithm CORE which can be very close to the result of algorithm ROCK while not considering many tuples as outliers.

Furthermore, algorithms ROCK and KModes are not stable to deal with different cases of data distribution. Sometimes algorithm ROCK even has the worse quality than algorithm KModes. More specifically, the best experimental result of Mushroom dataset by performing algorithm ROCK appears when we set the number of clusters to 5 (denoted by $|C| = 5$), which is larger than the number of class labels. Despite algorithm ROCK has a very high purity when it generates 21 clusters (which is the experimental result shown in [14]), the recall rate is small and therefore $Max(F_\beta)$ is not maximized in this case, meaning that the result is not the best when we consider the human viewpoint.

					$Max(F_\beta)$		
Data Name	#Classes	|R|	#Attributes	Missing	CORE	ROCK	KModes
Voting	2	435	16	Yes	0.86 $\|O\| = 3$	$0.87(\theta = 0.73)$ $\|O\| = 63$	0.85
mushroom	2	8124	22	Yes	0.87∗	$0.62(\theta = 0.55)$ $\|C\| = 5$	0.58
TAE	3	151	5	No	0.46∗	$0.21(\theta = 0.4)$ $\|O\| = 2$	0.35
Zoo	7	101	16	No	0.75	$0.77(\theta = 0.8)$ $\|O\| = 10$	0.75
Breast-Cancer	2	286	9	Yes	0.58∗	$0.41(\theta = 0.5)$ $\|O\| = 1$	0.51
Lymphography	4	148	18	No	0.42∗	$0.37(\theta = 0.6)$ $\|O\| = 11$	0.34

Table 3: The experimental results on publc domain real data.

5 Conclusion

In this paper, we proposed a high-quality algorithm, called CORE, for clustering categorical data. Specifically, in view of the phenomena from the observation of clustering categorical data, algorithm CORE is devised based on the proposed *correlated-force ensemble* technique. Many miscellaneous issues are also discussed to strengthen the practicability of algorithm CORE. As a result, the experiments show the outperformance of algorithm CORE in variant real datasets.

Acknowledgments

The authors are supported in part by the Ministry of Education Project No. 89-E-FA06-2-4, and the National Science Council Project No. NSC 91-2213-E-002-034 and NSC 91-2213-E-002-045, Taiwan, Republic of China. In addition, K.-T. Chuang was in part supported by the scholarship from Ericsson Taiwan.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[2] D. Barbara, Y. Li, and J. Couto. Coolcat: an entropy-based algorithm for categorical clustering. In *Proc. of ACM Int. Conf. on Information and Knowledge Management*, 2002.

[3] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is Nearest Neighbor Meaningful? *In Proc. of ICDT Conference*, 1999.

[4] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[5] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental Clustering and Dynamic Information Retrieval. *In Proc. of the ACM Symposium on Theory of Computing*, 1997.

[6] M.-S. Chen, J. Han, and P. S. Yu. Data Mining: An Overview from Database Perspective. *IEEE Trans. on Knowledge and Data Engineering*, Dec. 1996.

[7] J. C. Principe E. Gokcay. Information theoretic clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Feb. 2002.

[8] D. H. Fisher. Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning*, 1987.

[9] V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS-Clustering Categorical Data Using Summaries. *In Proc. of ACM SIGKDD*, 1999.

[10] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering Categorical Data: An Approach Based on Dynamical Systems. *In Proc of the 24th VLDB Conference*, 1998.

[11] A. Gluck and J. Corter. Information, uncertainty, and the utility of categories. In *Proc. of the Seventh Annual Conference of the Cognitive Science Society*, 1985.

[12] J. Gomez, D. Dsgupta, and O. Nasraoui. A new gravitational clustering algorithm. In *Proc. of the SIAM Int. Conf. on Data Mining (SDM)*, 2003.

[13] S. Guha, R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. *In Proc of ACM Int. Conf. on Management of Data*, 1998.

[14] S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. *In Proc. of the 15th Int. Conf. on Data Engineering*, 1999.

[15] Z. Huang. Extensions to the K-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery*, 1998.

[16] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[17] E. Rahm and H.-H. Do. Data cleaning: Problems and current approaches. *IEEE Bulletin of the Technical Committee on Data Engineering, 2000*, 2000.

[18] J. A. Rice. *Mathematical statistics and data analysis*. Duxbury Press, 1995.

[19] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proc. of ACM SIGKDD*, 2002.

[20] W. E. Wright. Gravitational clustering. *Pattern Recognition*, 9:151–166, 1977.