Extending $k$-Representative Clustering Algorithm with an Information Theoretic-based Dissimilarity Measure for Categorical Objects

Thu-Hien Thi Nguyen† Van-Nam Huynh†
†School of Knowledge Science
Japan Advanced Institute of Science and Technology, Japan
{nguyen.hien,huynh}@jaist.ac.jp

Abstract
This paper aims at introducing a new dissimilarity measure for categorical objects into an extension of $k$-representative algorithm for clustering categorical data. Basically, the proposed dissimilarity measure is based on an information theoretic definition of similarity introduced by Lin [15] that considers the amount of information of two values in the domain set. In order to demonstrate the efficiency of the extended $k$-representative algorithm with the new dissimilarity measure, we conduct a series of experiments on real datasets from UCI Machine Learning Repository and compare the result with several previously developed algorithms for clustering categorical data.

Keywords: Cluster Analysis, Categorical Data Clustering, K-means, Dissimilarity measure

1 Introduction
During the last two decades or so, data mining has emerged as a rapidly growing interdisciplinary field, which merges together databases, statistics, machine learning and other related areas in order to extract useful knowledge from data [1]. Cluster analysis or simply clustering is one of fundamental tasks in data mining that aims at grouping a set of data objects into multiple clusters, such that objects within a cluster are similar one another, yet dissimilar to objects in other clusters. Dissimilarities and similarities between objects are assessed based on those attribute values describing the objects and often involve distance measures.

Typically, objects can be considered as vectors in $n$-dimensional space, where $n$ is the number of features. When objects are described by numerical features, the distance measure based on geometric concept such as Euclid distance or Manhattan distance. But these geometric distance measures are not applicable for categorical data which contains values from gender, locations, etc. Recently, clustering data with categorical attributes have gained increasing attention [9; 7; 10; 8; 4; 5]. As for categorical data, the comparison measure is most naturally used [4]. However, this metric does not distinguish between the different values taken by the attribute, since we only measure the equality between pair of values, as argued in [16].

In this paper we propose a new approach to compute the dissimilarity between values of a categorical variable and apply it to $k$-representative clustering of categorical data. The solution is to consider the amount of information between values in the domain set of a categorical attribute. The closer the amount of information between values are, the more similarities they share [15].

The remainder of this paper is organized as follows: Section 2 briefly describes related work. Section 3 recalls the notations and definitions used in this paper. Section 4 deals with the function of $k$-means algorithm. The proposed method is discussed in Section 5. In Section 6, we present the results of the experiments on data sets. Finally, Section 7 concludes the paper.

2 Related work
Probably, the $k$-means clustering [2] is the most well-known approach for clustering data sets with numerical attributes. It is a traditional partitioning based approach which starts with $k$ random centroids and the centroids are updated iteratively by computing the average of the numerical features in each cluster. Each observation or object is assigned to clusters based upon the nearest distance to the means of the clusters. The iteration continues until the assignment is stable, that is, the clusters formed in the current round are the same as those formed in the previous round. The $k$-means is very popular due to its ease of implementation, linear time complex-
ity in size of the data, and almost surely convergence to local optima [3]. However, in real life many data set are categorical. In such situations, \textit{k}-means cannot be applied.

Few algorithms have been proposed in recent years in order to overcome the numerical-only limitation of \textit{k}-means, like \textit{k}-modes [5], \textit{k}-representative [13]. Huang proposed \textit{k}-modes, an extension of \textit{k}-means, which is developed to cluster categorical data. Basically, the \textit{k}-modes algorithm uses a simple matching similarity measure for categorical objects, and modes instead of means for cluster centers. Mode is a data point, in which the value of each attribute is assigned the most frequent value in the attribute’s domain set. Further, Huang also combined the \textit{k}-modes algorithm with \textit{k}-means algorithm in order to deal with mixed numerical and categorical databases. These extensions allow us to efficiently cluster very large data sets from real world applications. However, the performance of \textit{k}-mode algorithm depend strongly on the selection of the modes during the clustering process. In an attempt to overcome this drawback, San et al. [13] introduced a new dissimilarity measure to the \textit{k}-modes clustering process to improve the accuracy that resulted in the so-called \textit{k}-representative algorithm. The main idea of \textit{k}-representative algorithm is to use the relative attribute frequencies of the cluster modes in the similarity measure in the \textit{k}-modes objective function. It has been shown that the \textit{k}-representative algorithm is very effective [14].

Along with trying to adapt \textit{k}-means for clustering categorical objects, other approaches such as ROCK [10], COOLCAT [12], LIMBO [11] have been developed. Particularly, ROCK [10] uses the user-defined threshold of Jaccard similarity measure between the two objects to define the links between them. Then it tries to optimize cluster objective function with respect to the number of links between the cluster members. The basic algorithm is hierarchical and not suitable for large datasets due to computational complexity. COOLCAT is information theory based approach. It tries to optimize the global utility function based on entropy criterion. The original approach is highly dependent on the initial conditions. LIMBO is a scalable hierarchical categorical clustering algorithm built on the Information Bottleneck framework. As a hierarchical algorithm, LIMBO is not as fast as partitional methods.

In the experiment phase, we compare the results from our proposed approach with both other extensions of \textit{k}-means algorithm and other approaches for categorical clustering.

3 Definitions and Notation

We assume that the set of objects to be clustered is stored in a dataset \( D \) defined by a set of attributes \( A_1, \ldots, A_m \) with domains \( D_1, \ldots, D_m \), respectively. Each object in \( D \) is represented by a tuple \( t \in D_1 \times \ldots \times D_m \). Regarding the clustering problem discussed in this paper, we consider two types of data: \textit{numeric} and \textit{categorical}. The domain of numerical attributes consists of continuous real values. Thus, the distance measure based on geometric concept such as the Euclidean distance or Manhattan distance can be used. A domain \( D_i \) is defined as categorical if it is finite and unordered, that only a comparison operation is allowed in \( D_i \). It means, for any \( x, y \in D_i \), we have either \( x = y \) or \( x \neq y \).

For simplicity, we represent each data object \( X \) as a tuple \( (x_1, \ldots, x_m) \in D_1 \times \ldots \times D_m \).

4 \textit{k}-means Algorithm

The \textit{k}-means algorithm [2] is one of the most popular algorithm in \textit{partitional} or \textit{non-hierarchical} clustering methods. Given a set \( D \) of \( n \) numerical data objects, a natural number \( k \leq n \), and a distance measure \( d \), the \textit{k}-means algorithm searches for a partition of \( D \) into \( k \) non-empty disjoint clusters that minimizes the overall sum of the squared distances between data objects and their cluster centers. The problem can be formulated in terms of an optimization problem as follow:

Minimize

\[
P(W, Q) = \sum_{l=1}^{k} \sum_{i=1}^{n} w_{i,l} d(X_i, Q_l)
\]

subject to

\[
\sum_{l=1}^{k} w_{i,l} = 1, \quad 1 \leq i \leq n,
\]

\[
w_{i,l} \in \{0, 1\}, \quad 1 \leq i \leq n, 1 \leq l \leq k,
\]

where \( W = [w_{i,l}]_{n \times k} \) is a partition matrix \( (w_{i,l}) \) take value 1 if object \( X_i \) is in cluster \( C_l \), and 0
otherwise), \( Q = \{ Q_1, \ldots, Q_k \} \) is a set of cluster centers, and \( d((\cdot), (\cdot)) \) is the squared Euclidean distance between two objects.

The problem \( P \) can be solved by iteratively solving two problems:

- Fix \( Q = \tilde{Q} \) then solve the reduced problem \( P(W, \tilde{Q}) \) to find \( W \).
- Fix \( W = \tilde{W} \) then solve the reduced problem \( P(W, Q) \).

Basically, the \( k \)-means algorithm iterates through a three-step process until \( P(W, Q) \) converges to some local minimum:

1. Select an initial \( Q^{(0)} = Q_1^{(0)}, \ldots, Q_k^{(0)} \), and set \( t = 0 \).
2. Keep \( Q^{(t)} \) fixed and solve \( P(W, Q^{(t)}) \) to obtain \( W^{(t)} \), that is, having the cluster centers, we then assign each object to the cluster of its nearest cluster center.
3. Keep \( W^{(t)} \) fixed and generate \( Q^{(t+1)} \) such that \( P(W^{(t)}, Q^{(t+1)}) \) is minimized, that is, construct new cluster centers according to the current partition.

In the case of convergence or if a given stopping criterion is fulfilled, output the result and stop. Otherwise, set \( t = t + 1 \) and go to step 2.

In numerical clustering problems, the Euclidean norm is often chosen as a natural distance measure in the \( k \)-means algorithm. With this distance measure, we calculate the partition matrix in step 2 as below, and the cluster center is computed by the mean of cluster’s objects.

\[
\text{if } d(X_i, C_l) \leq d(X_i, C_p) \text{ then } w_{i,l} = 1, \text{ and } w_{i,p} = 0, \quad \text{for } 1 \leq p \leq k, p \neq l
\]  

(3)

It was also shown in [4] that the \( k \)-means method can be extended to categorical data by using a simple matching distance measure for categorical objects with the most frequent values to define the “cluster centers” called modes. Let \( X, Y \) are two categorical objects in \( D \), with \( X = (x_1, \ldots, x_m) \) and \( Y = (y_1, \ldots, y_m) \). The dissimilarity between \( X \) and \( Y \) can be computed by the total matching of the corresponding attribute values of the two objects. Formally,

\[
d(X, Y) = \sum_{i=1}^{m} \delta(x_i, y_i)
\]

(4)

where

\[
\delta(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i; \\ 1 & \text{if } x_i \neq y_i. \end{cases}
\]

Given a cluster \( X_1, \ldots, X_p \) of categorical objects, with \( X_i = (x_{i,1}, \ldots, x_{i,m}) \), \( 1 \leq i \leq p \), its mode \( Q = (q_1, \ldots, q_m) \) is defined by assigning \( q_j \), \( 1 \leq j \leq m \), the value most frequently appeared in \( \{ x_{1,j}, \ldots, x_{p,j} \} \). With these modifications, Huang [5] proposed a \( k \)-modes algorithm that mimics the \( k \)-means method to cluster categorical data. However, it should be emphasized that, by definition, the mode of a cluster is not in general unique. This makes the algorithm unstable depending on mode selection during the clustering process.

In 2004, San et al. [13] proposed a new alternative algorithm, \( k \)-representative, trying to overcome the drawback of the \( k \)-modes. The modes are replaced by “representatives”. Given a cluster \( C = X_1, \ldots, X_p \) of categorical objects, with

\[
X_i = (x_{i,1}, \ldots, x_{i,m}), \quad 1 \leq i \leq p,
\]

denote by \( D_j \) the domain set of categorical attribute \( A_j \). Then the representative of \( C \) is defined by \( Q = (q_1, \ldots, q_m) \), with

\[
q_j = \{ (c_j, f_{c_j}) | c_j \in D_j \}
\]

(5)

where \( f_{c_j} \) is the relative frequency of category \( c_j \) in \( C \), i.e., \( f_{c_j} = n_{c_j}/p \), where \( n_{c_j} \) is the number of objects in \( C \) having value \( c_j \) at attribute \( A_j \).

The dissimilarity between a categorical object \( X = (x_1, \ldots, x_m) \) and the cluster representative \( Q \) is calculated based on simple matching (cf.(4)) as follows

\[
d(X, Q) = \sum_{j=1}^{m} \sum_{c_j \in D_j} f_{c_j} \delta(x_j, c_j)
\]

(6)

The formulation shows that the dissimilarity \( d(X, Q) \) is mainly dependent on the relative frequencies of categorical values within the cluster and the simple matching between categorical value.

5 Proposed Approach

5.1 Proposed dissimilarity measure

In this paper, instead of using simple matching measure, we propose a new dissimilarity measure which is based on an information-theoretic definition of similarity [15]. For example, \( x \) and \( y \) are values from categorical values set of attribute \( A_j \). We can describe \( x \) and
y with the proposition like: value\_\text{A}_j(x) and value\_\text{A}_j(y). The proposition that states the commonality between x and y is “value\_\text{A}_j(x) and value\_\text{A}_j(y)”.

In information theory, we can calculate the information contained in a statement by the negative logarithm of the probability of the statement. From that, the similarity between x and y is measured by the ratio between the amount of information needed to state the commonality of x and y and the information needed to fully describe what x and y are. Specifically, we define the measure as,

\[
\delta(x, y) = 1 - \frac{2 \times \log f_{x,y}}{\log f_x + \log f_y},
\]

where \(f_x, f_y\) are the frequency of the value x and y, respectively at the attribute \(A_j\) in \(D\). \(f_{x,y}\) is the frequency that an object in \(D\) received value either x or y at attribute \(A_j\). We can see that, if x and y are identical then \(\delta(x, y) = 0\), otherwise \(0 < \delta(x, y) < 1\).

5.2 \(k\)-Representatives Algorithm with proposed dissimilarity measure

With the modification above, we formulate the problem of clustering categorical data in much the same way as in the \(k\)-representative algorithm proposed in [13].

Assume that we have a data set \(D = X_1, \ldots, X_n\) of categorical objects to be clustered, where each object \(X_i = (x_{i,1}, \ldots, x_{i,m})\), \(1 \leq i \leq n\) is described by \(m\) categorical attributes. Mathematically, the problem can be stated as: Minimize

\[
P(W, Q) = \sum_{l=1}^{k} \sum_{i=1}^{n} w_{i,l} d(X_i, Q_l)
\]

subject to

\[
\sum_{l=1}^{k} w_{i,l} = 1, \quad 1 \leq i \leq n,
\]

\[
w_{i,l} \in \{0, 1\}, \quad 1 \leq i \leq n, 1 \leq l \leq k,
\]

where \(W = [w_{i,l}]_{n \times k}\) is a partition matrix, \(Q = \{Q_1, \ldots, Q_k\}\) is a set of cluster representatives, and \(d(., .)\) is the dissimilarity measure defined by (6) with the modified \(\delta(., .)\) in (7). The algorithm is processed as:

1. Randomly initialize a \(k\)-partition of \(D\).
2. Calculate the representatives for each cluster.
3. Reassign objects to clusters regarding the dissimilarity between objects and cluster representatives. When finish, recalculate the representatives.
4. Repeat step 3 until the partition has no change or the number of iterations is beyond user-defined threshold.

6 Experiments and Results

In this section we present a comprehensive evaluation of our approach. In terms of extension of \(k\)-means algorithm for categorical objects, our approach is compared with \(k\)-representative algorithm. And we also want to compare the performance of our approach with other methods for clustering categorical data such as: ROCK [10] and LIMBO [11].

6.1 Datasets

For the evaluation, we used real world data sets downloaded from the UCI Machine Learning Repository [18]. The main characteristics of the datasets are summarized in Table 1. These datasets are chosen to test our algorithm because of their public availability and since all attributes can be treated as categorical ones.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Number of Instances</th>
<th>Number of Attributes</th>
<th>Number of Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>1728</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Soybean (Small)</td>
<td>47</td>
<td>35</td>
<td>4</td>
</tr>
<tr>
<td>Soybean (Large)</td>
<td>683</td>
<td>35</td>
<td>19</td>
</tr>
<tr>
<td>Votes</td>
<td>435</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>Nursery</td>
<td>12960</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Mushroom</td>
<td>8124</td>
<td>22</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1: Datasets Summarization

6.2 Clustering Evaluation Metrics

Evaluating the clustering quality is often a hard and subjective task [16]. Generally, objective functions in clustering are purposely designed so as to archive high intra-cluster similarity and low inter-cluster similarity. This can be viewed as an internal criterion for the quality of a clustering. However, as observed in the literature, good
scores on an internal criterion do not necessarily translate into good effectiveness in an application. Here, by the same way as in [17], we use three external criteria to evaluate the results: Purity, Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI). These methods make use of the original class information of each object and the cluster to which the same objects have been assigned to evaluate how well the clustering result matches the original classes.

We denote by $C = \{C_1, \ldots, C_J\}$ the partition of the dataset built by the clustering algorithm, and by $P = \{P_1, \ldots, P_I\}$ the partition inferred by the original classification. $J$ and $I$ are respectively the number of clusters $|C|$ and the number of classes $|P|$. We denote by $n$ the total number of objects.

Purity is a simple and transparent evaluation measure. To compute purity, each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned objects and dividing by the number of objects in the dataset. High purity is easy to achieve when the number of clusters is large. Thus, we cannot use purity to trade off the quality of the clustering against the number of clusters.

$$Purity(C, P) = \frac{1}{n} \sum_j \max_i |C_j \cap P_i|$$  \hspace{1cm} (10)

The second metric (NMI) provides an information that is independent from the number of clusters [19]. This measure takes its maximum value when the clustering partition matches completely the original partition. NMI is computed as the average mutual information between any pair of clusters and classes

$$NMI(C, P) = \frac{\sum_{i=1}^I \sum_{j=1}^J |C_j \cap P_i| \log \frac{n|C_j \cap P_i|}{|C_j||P_i|}}{\sqrt{\sum_{j=1}^J |C_j| \log \frac{|C_j|}{n} \sum_{i=1}^I |P_i| \log \frac{|P_i|}{n}}}$$  \hspace{1cm} (11)

The third metric is the adjusted Rand index [20]. Let $a$ be the number of object pairs belonging to the same cluster in $C$ and to the same class in $P$. This metric captures the deviation of $a$ from its expected value corresponding to the hypothetical value of $a$ obtained when $C$ and $P$ are two random, independent partitions. The expected value of $a$ denoted by $E[a]$ is computed as follows:

$$E[a] = \frac{\pi(C)\pi(P)}{n(n-1)/2}$$  \hspace{1cm} (12)

where $\pi(C)$ and $\pi(P)$ denote respectively the number of object pairs from the same clusters in $C$ and from the same class in $P$. The maximum value for $a$ is defined as:

$$\max(a) = \frac{1}{2}(\pi(C) + \pi(P))$$  \hspace{1cm} (13)

The agreement between $C$ and $P$ can be estimated by the adjusted rand index as follows:

$$ARI(C, P) = \frac{a - E[a]}{\max(a) - E[a]}$$  \hspace{1cm} (14)

when $ARI(C, P) = 1$, we have identical partitions.

### 6.3 Results

We run the experiments on a Mac with a 3.66GHz Intel QuadCore processor, 8GB of RAM running Mac OSX 10.8. The performance of three evaluation metrics are calculated by the average after running the algorithm 1000 times on each dataset. For the performance result of ROCK and LIMBO, we obtained from the experiments in [17]. In the tables 2, 3 and 4 we report the results of the comparative evaluation against ROCK, LIMBO and k-representative. From the results, we can see that our approach and k-representative are very much close to each other in terms of performances with almost datasets, except the nursery dataset where our approach obtain slightly better results in all three criteria. It can be seen that, ROCK and LIMBO are not stable enough. In some case their performance are good enough, but in other case the performance are far from the results of our approach. For example, ROCK is worst with mushroom dataset but quite good with car dataset or vote dataset. And LIMBO are good enough, but in other case the performances are quite stable. Regarding the purity criteria, we got the best results on 3 out of 4 datasets. And for two remain metrics, we also got best results in 2 out of 4 datasets and the other one is close to the best. Especially, with car dataset, our modified $k$-representative algorithm outperform ROCK and LIMBO in all evaluation metrics.

### 7 Conclusion

In this paper, we proposed a new approach to define the dissimilarity for categorical data and
applied the method to an alternative algorithm that mimics the $k$-means method in clustering categorical data. The performance of new approach is compared with other categorical clustering algorithms. The experimental results have shown that the proposed approach gets good results and is more stable than ROCK and LIMBO algorithm. In addition, it is also comparable or slightly better to the original $k$-representative algorithm. However, the improvement is less than what we expected. As for future work, we would like to apply another way to assign the dissimilarity between two matches of categorical attribute values (in the current measure, matches are uniformly given the maximum value as 1). For example, we could give lower weight to matches on frequent values, and higher weight to matches on infrequent values or rare values. It is also of interest to extend this technique to the problem of clustering for mixed datasets as well as to the problem of fuzzy clustering for categorical data.

References


[9] V. Ganti, J. Gehrke, Raghu Ramakrishnan. CATUS - Clustering categorical data using

### Table 2: Purity results

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Proposed algorithm</th>
<th>$k$-representative</th>
<th>ROCK</th>
<th>LIMBO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>0.71</td>
<td>0.7</td>
<td>0.7</td>
<td>0.44</td>
</tr>
<tr>
<td>Soybean (Small)</td>
<td>0.93</td>
<td>0.93</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Soybean (Large)</td>
<td>0.56</td>
<td>0.57</td>
<td>0.44</td>
<td>0.56</td>
</tr>
<tr>
<td>Votes</td>
<td>0.87</td>
<td>0.88</td>
<td>0.84</td>
<td>0.87</td>
</tr>
<tr>
<td>Nursery</td>
<td>0.46</td>
<td>0.44</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Mushroom</td>
<td>0.78</td>
<td>0.77</td>
<td>0.5</td>
<td>0.89</td>
</tr>
</tbody>
</table>

### Table 3: NMI results

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Proposed algorithm</th>
<th>$k$-representative</th>
<th>ROCK</th>
<th>LIMBO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>0.12</td>
<td>0.11</td>
<td>0.036</td>
<td>0.014</td>
</tr>
<tr>
<td>Soybean (Small)</td>
<td>0.93</td>
<td>0.93</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Soybean (Large)</td>
<td>0.7</td>
<td>0.7</td>
<td>0.63</td>
<td>0.75</td>
</tr>
<tr>
<td>Votes</td>
<td>0.5</td>
<td>0.49</td>
<td>0.34</td>
<td>0.44</td>
</tr>
<tr>
<td>Nursery</td>
<td>0.11</td>
<td>0.094</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Mushroom</td>
<td>0.35</td>
<td>0.35</td>
<td>0.057</td>
<td>0.57</td>
</tr>
</tbody>
</table>
Table 4: Adjusted Rand Index results

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Proposed algorithm</th>
<th>$k$-representative</th>
<th>ROCK</th>
<th>LIMBO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>0.06</td>
<td>0.052</td>
<td>0.004</td>
<td>0.037</td>
</tr>
<tr>
<td>Soybean (Small)</td>
<td>0.88</td>
<td>0.88</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Soybean (Large)</td>
<td>0.38</td>
<td>0.37</td>
<td>0.17</td>
<td>0.44</td>
</tr>
<tr>
<td>Votes</td>
<td>0.56</td>
<td>0.56</td>
<td>0.46</td>
<td>0.55</td>
</tr>
<tr>
<td>Nursery</td>
<td>0.075</td>
<td>0.058</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Mushroom</td>
<td>0.39</td>
<td>0.39</td>
<td>-0.001</td>
<td>0.61</td>
</tr>
</tbody>
</table>


