Relevant Overlapping Subspace Clusters on Categorical Data

Xiao He[1], Jing Feng[1], Bettina Konte[1], Son T.Mai[1], Claudia Plant[2]
[1]: University of Munich, [2]: Helmholtz Zentrum München, Technische Universität München
{he, feng, konte, mtson}@dbs.ifi.lmu.de, claudia.plant@helmholtz-muenchen.de

ABSTRACT

Clustering categorical data poses some unique challenges: Due to missing order and spacing among the categories, selecting a suitable similarity measure is a difficult task. Many existing techniques require the user to specify input parameters which are difficult to estimate. Moreover, many techniques are limited to detect clusters in the full-dimensional data space. Only few methods exist for subspace clustering and they produce highly redundant results. Therefore, we propose ROCAT (**R**elevant **O**verlapping Subspace Clusters on **Cat**egorical Data), a novel technique based on the idea of data compression. Following the Minimum Description Length principle, ROCAT automatically detects the most relevant subspace clusters without any input parameter. The relevance of each cluster is validated by its contribution to compress the data. Optimizing the trade-off between goodness-of-fit and model complexity, ROCAT automatically determines a meaningful number of clusters to represent the data. ROCAT is especially designed to detect subspace clusters on categorical data which may overlap in objects and/or attributes; i.e. objects can be assigned to different clusters in different subspaces and attributes may contribute to different subspaces containing clusters. ROCAT naturally avoids undesired redundancy in clusters and subspaces by allowing overlap only if it improves the compression rate. Extensive experiments demonstrate the effectiveness and efficiency of our approach.

Categories and Subject Descriptors

H.2.8 [**Database applications**]: Data Mining

Keywords

Relevant Subspace Clustering; Categorical Data; Minimum Description Length

KDD'14, August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623652.

1. INTRODUCTION

In many applications, ranging from social media to biomedicine, large categorical data sets are collected. The unique characteristic of categorical data is that the values of an attribute do not have any order. For example the attribute *genotype* having four values AA, Aa, aA and aa, where capital A represents the dominant normal variant of a gene and lowercase a the recessive version. There is no implicit order or quantitative spacing between the different categories.

To explore a large categorical data set, clustering is in principle very promising. Among the most successful approaches to unsupervised data mining, clustering aims at finding a natural partitioning of a data set into groups called clusters which represent the major patterns in the data. However, there are several special challenges associated with clustering moderate to high-dimensional categorical data:

1) Many existing algorithms require the user to choose a similarity metric and/or input parameters which are difficult to estimate, e.g. k-Modes [14], COOLCAT [5] and ROCK [13]. In comparison to numerical data where Minkowski distances are wide-spread and well-explored, the choice of a suitable similarity measure for categorical data is much more difficult: In a comparative survey [6], Boriah et al. studied the properties of 14 similarity measures and concluded that a suitable choice requires deep knowledge on the envisaged data mining task and the special characteristics of the data set to be analyzed. The same holds for input parameters like the number of clusters k in k-Modes [14], COOLCAT [5], SUBCAD [9], or the similarity threshold in ROCK [13].

2) Most techniques for clustering categorical data are limited to detect clusters in the full-dimensional space. For numerical data, the effects of the so-called *curse of dimensionality* have been extensively studied and many specialized techniques for clustering moderate to high-dimensional data have been proposed, for a survey see e.g. [16]. For categorical data, fewer approaches have been proposed, e.g. CACTUS [10], SUBCAD [9], and CLICKS [26], most of them associated with the problems mentioned above.

3) These methods are either partition-based (e.g. SUBCAD) or producing large redundancies (e.g. CLICKS). STATPC [19] and RESCU [20] are proposed to find relevant non-redundant subspace clusters but are applicable to numerical data only. Detecting relevant overlapping subspace clusters on categorical data is an open research question.

To address these challenges, we propose a novel approach ROCAT (Relevant Overlapping Subspace Clusters on CATegorical data) combining the following benefits:

1. **Data compression as an intuitive notion of similarity.** Relating clustering to data compression, ROCAT considers the co-compressibility of the objects inside a cluster as one major aspect to evaluate the cluster quality. Thereby, ROCAT does not require the user to choose a similarity measure to quantify the pair-wise similarity among categorical data objects.

2. **Parameter-free detection of clusters.** Following the Minimum Description Length principle [22], co-compressibility is not the only aspect of cluster quality in ROCAT. We additionally consider the code length specifying the complexity of the clustering model and aim at minimizing both parts. Therefore, ROCAT is fully-automatic without requiring any parameters.

3. **Relevant overlapping subspace clusters.** The coding scheme of ROCAT allows overlapping in both objects and attributes set, but punishes redundancies. Therefore ROCAT finds the most relevant overlapping subspace clusters in the sense that they contribute to an effective compression of the data.

4. **Flexibly handling outliers.** ROCAT supports noise objects and noise attributes which are flexibly identified during the clustering process.

5. **Efficiency.** ROCAT scales linearly in data size.

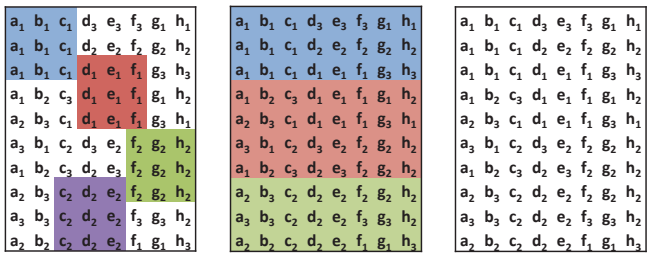
The remainder of this paper is organized as follows: In the following section, we elaborate our optimization goal. Section 3 presents the algorithm in detail. Section 4 contains an extensive experimental evaluation. Section 5 briefly discusses related work and Section 6 concludes the paper.

2. OPTIMIZATION GOAL COMPRESSION

In this section, we elaborate how a subspace clustering can be used to effectively compress a categorical data set. The basic idea is that objects inside a cluster can be compactly represented by joint coding in the corresponding subspace. Since subspace clusters may overlap, we validate the relevance of each cluster by its contribution to compress the data. Following the Minimum Description Length (MDL) principle [22], the clustering is regarded as a model for compression. The better the data fits to the model, the better is the compression rate since we only need to encode the deviations of the data from the model. In addition to the data, we also need to encode the model itself, which avoids overly complex models and naturally balances goodness-of-fit.

Figure 1 depicts an example of how we use compression to evaluate a clustering of categorical data. The data is represented by a matrix with 10 rows and 8 columns, each row represents 40 data objects and each column is an attribute. Figure 1(a) indicates a subspace clustering with 4 subspace clusters marked as colored squares. The clusters share the homogeneous data in the corresponding subspace, while the white area represents heterogeneous data, i.e. objects have arbitrary categories of the corresponding attributes. It costs 6076.5 bits to compress the data with the proposed coding scheme. Figure 1(b) and 1(c) depict a full-dimensional clustering and no clustering, where we need 6147.1 bits and 6670.9 bits to represent the whole data sets.

In the following, we firstly provide the necessary definitions and then propose a MDL-based coding scheme, which is specially designed for clustering categorical data.



(a)Subspace Clustering (b)Full-D Clustering (c)No Clustering

Figure 1: Using compression to evaluate clustering.

2.1 Notations

DEFINITION 1. *A **Categorical Data Set** is defined as $D = (X, V)$ with N objects and M attributes. $A_1, ..., A_M$ denote a set of categorical attributes and $V_1, ..., V_M$ a set of domains, where $V_j = \{V_{j_1}, ..., V_{j_m}\}$ is the domain for attribute A_j. $X \in N \times M$ is a matrix storing the categorical value $X(i, j)$ of object i in attribute A_j.*

DEFINITION 2. *A **Subspace Cluster** $C_i = (X_i, V_i)$ is a subset of the data set $D = (X, V)$, where X_i is a sub-matrix of X and V_i is a subset of V.*

DEFINITION 3. *A **Pure Subspace Cluster** is a Subspace Cluster where the objects share the same value in all its attributes.*

DEFINITION 4. *A **Non-Clustered Area** S of data D modeled by subspace clusters $C_1, C_2, ..., C_K$ contains all the entries in matrix X that are not in any sub-matrix X_i, the white area in Figure 1(a).*

2.2 Coding Scheme

Following the concept of MDL principle, the quality of a model is provided by Eq. (1), where $L(H)$ denotes the cost for coding the model and $L(D|H)$ represents the cost of describing the data D under the model H.

$$L(D, H) = L(D|H) + L(H). \quad (1)$$

The model H contains K subspace clusters $C_1, C_2, ..., C_K$ and the non-clustered area S. According to our definition of a Subspace Cluster, $C_1, C_2, ..., C_K$ might overlap in both points and attributes set. The description of data D under the model H is provided by describing $C_1, C_2, ..., C_K$ and S separately. Therefore, $L(D|H)$ consists of two parts: the costs for the clusters and the non-clustered area.

$$L(D|H) = \sum_{i=1}^{K} CC_v(C_i) + CC_v(S). \quad (2)$$

In order to quantify the description length of a subspace cluster C_i or the non-clustered area S, we need to agree on an encoding scheme for C_i and S. For each cluster C_i, we encode the corresponding data sub-matrix X_i of C_i column by column. Specifically, for each assigned attribute A_j of cluster C_i, we calculate the probabilities for all categories in attribute A_j. Then any lossless coding method can be used to compress the column of attribute A_j in matrix X_i, i.e. Huffman coding. Practically we only need the coding length for evaluation, but not the true bits stream. Besides, lossless coding methods are lower bounded by the Shannon entropy.

Therefore, we suggest to calculate the coding length of an attribute A_j in cluster C_i with categories $V_j = \{V_{j_1}, ..., V_{j_m}\}$ by the Shannon Entropy, which is defined as:

$$Entropy(P) = -\sum_{k=V_{j_1}}^{V_{j_m}} \cdot P_k \cdot log_2 P_k, \quad (3)$$

$P = P_{V_{j_1}}, ...P_{V_{j_m}}$ are the probabilities of the categories in attribute A_j and cluster C_i, where $P_k = \frac{|X_i(:,j)==k|}{|C_i.obj|}$ and $|\cdot|$ is the number of entries in a set.

Then the coding cost for cluster C_i is provided as:

$$CC_v(C_i) = \sum_j |C_i.obj| \cdot Entropy(P). \quad (4)$$

The coding cost for the non-clustered area $CC_v(S)$ is calculated analogously to Eq. (4).

In addition to the data, we also need to describe the model itself $L(H)$. The model H contains the clustering assignments and probabilities that are used to encode C_i and S in Eq. (4). We need to describe both the object assignments CC_o and the attribute assignments CC_a to encode the clustering assignments for each subspace cluster C_i. In addition, we need to encode the probabilities used to describe data $L(D|H)$ since they are essential for lossless decoding. These probabilities are encoded as parameters CC_r.

$$L(H) = \sum_{i=1}^{K}(CC_o(C_i)+CC_a(C_i)+CC_r(C_i))+CC_r(S). \quad (5)$$

Specifically, we describe the clustering assignments with object ID tables and attribute ID tables. The object ID table for a subspace cluster is a length N binary table. Objects are assigned 1 if they belong to the cluster, and 0 otherwise. As before, we suggest to encode the tables using an optimal Shannon code. The coding cost for an object ID table of C_i is calculated by its Shannon entropy:

$$CC_o(i) = -N \cdot (p(i) \cdot log_2 p(i) + n(i) \cdot log_2 n(i)), \quad (6)$$

where $p(i) = \frac{|C_i.obj|}{N}$ is the percentage of 1s, $n(i) = 1 - p(i)$ is the percentage of 0s. The coding cost for the attribute ID table is derived analogously to Eq. (6).

We encode the probabilities used in Eq. (4) as parameters. Following [22], the cost for the probabilities can be approximated by:

$$CC_r(i) = 0.5 \cdot |Param| \cdot log_2 |C_i.obj|, \quad (7)$$

where $|Param|$ is the number of parameters or probabilities. For each assigned attribute A_j of cluster C_i we need to encode $|V_j|$ probabilities. Therefore, $|Param| = \sum_j (|V_j|)$. The parameter cost for the non-clustered part $CC_r(S)$ is calculated analogously to Eq. (7).

The relevance of each cluster is validated by its contribution to compress the data. Thus the proposed coding scheme is perfectly suitable to evaluate relevant overlapping subspace clusters on categorical data. Firstly, it allows overlapping clusters in both objects and attributes set, but punishes those redundancies, since the overlapping parts will be encoded twice. Secondly, it avoids too complex models (too many small clusters) by encoding the model itself (clustering assignments and the probabilities), thus large informative clusters are preferred. In summary, clustering with the most relevant overlapping subspace clusters will achieve a lower coding cost under the proposed coding scheme.

3. ALGORITHM

In this section, we present an effective and efficient algorithm to identify the most relevant overlapping subspace clusters. Our optimization goal is to find the clustering model that best describes the categorical data set under the proposed coding scheme in Section 2.

3.1 Minimum Coding Problem

The proposed coding scheme does not specify how to find a good clustering; it can only say which of two clusterings is better. The problem, which we call the Minimum Coding Problem in the following, can be modeled as finding sub-matrices (Subspace Cluster) that allow the highest compression with respect to the proposed coding scheme. Given a data set D with N objects and M attributes, there are $I = (\sum_i^N \binom{N}{i}) \cdot (\sum_j^M \binom{M}{j})$ possible sub-matrices, further there are $\sum_i^I \binom{I}{i}$ possible clusterings with different combinations of sub-matrices. Obviously, an naive exhaustive search for the optimal result is infeasible even for a small data set, since the number of candidates $|I|$ is exponential to M and N. Even for the case that $|I|$ is polynomial to M and N, the Minimum Coding Problem is a NP-hard problem.

Minimum Coding Problem is NP-hard. The Set Covering Problem is known to be NP-hard [11]. Given a set of elements U and a set E of n sets, the Set Covering Problem finds smallest subsets of E whose union cover all the elements in U. The Minimum Coding Problem aims at finding sub-matrices of a data matrix X that cover all the entries of X, but uses a different kind of cost function, i.e. the proposed coding scheme. In the case that $|I|$ is polynomial and except of using a different cost function, the Minimum Coding Problem is equivalent to the Set Covering Problem or the Weighted Set Covering Problem. Since the proposed coding function can be calculated in polynomial time, the Minimum Coding Problem is NP-hard as well.

In summary, the Minimum Coding Problem is so difficult that we need an efficient and effective heuristic algorithm to achieve a local optimal result.

3.2 Algorithm ROCAT

The best-possible polynomial time approximation algorithm for the Set Cover Problem is the greedy algorithm [17]. At each stage, the set that contains the largest number of uncovered elements is selected. However, the greedy algorithm can not be used directly to solve the Minimum Coding Problem due to the following reasons. Firstly, the Minimum Coding Problem uses a different cost function, thus including the set that contains the largest number of uncovered elements may not reduce the proposed coding function. Secondly, the number of candidates $|I|$ is exponential to M and N, which makes the greedy algorithm exponential as well.

The proposed algorithm ROCAT is based on the greedy idea as well, but some essential modifications are made to solve the above two problems. Firstly, we need to iteratively include the Subspace Cluster that reduces the overall cost under the proposed coding function. Secondly, we need to reduce the number of candidate Subspace Clusters for greedy selection. Eq. (4) shows that including large Pure Subspace Clusters will lead to a reduction of the coding cost. Additionally, searching for large Pure Subspace Clusters will reduce the candidate space to polynomial as well. Therefore, we focus on selecting the Pure Subspace Clusters at first, then post-process them to get the final Subspace Clusters.

Algorithm 1 ROCAT

Input: Data set $D = (X, V)$
Output: Subspace clusters list $SubClus$

//Searching phase
$SubClus = \emptyset$; Queue $queue = \emptyset$; $queue$.Push(D);
while $queue \neq \emptyset$ **do**
  $Curr = queue$.Pop;
  $C =$ **FindBestPure**($Curr, SubClus$);
  if Eq. (1) decreases with $SubClus \cup C$ **then**
    $SubClus$.Add(C);
    $queue$.PushAll(**SplitSpace**($Curr, C$));
  end if
end while

//Combining phase
Priority queue $Pairs = \emptyset$;
for Each pair of clusters $C_i, C_j \in SubClus$ **do**
  $Overlap = |C_i.obj \cap C_j.obj| \cdot |C_i.att \cap C_j.att|$;
  If $Overlap > 0$, $Pairs$.Push(C_i, C_j);
end for
while $Pairs \neq \emptyset$ **do**
  Process $Pairs$.Pop as shown in Figure 3;
  Choose one process with minimum Eq. (1);
end while

//Reassigning phase
while Convergence **do**
  for Each cluster $C_i \in SubClus$ **do**
    Find all objects set O with same value in $C_i.att$;
    Assign or Remove O to $C_i.obj$ based on Eq.(1);
  end for
  for Each cluster $C_i \in SubClus$ **do**
    If $C_i.obj$ changed, Re-select $C_i.att$ based on Eq.(1);
  end for
end while

return $SubClus$

The found Pure Subspace Clusters can overlap and exhibit redundancies. Therefore, during post-processing we firstly combine or split them to remove redundancies, then refine the results by locally modifying clustering assignments.

More precisely, there are three phases in ROCAT: Searching, Combining and Reassigning. Firstly, we iteratively include large Pure Subspace Clusters if the coding cost can be reduced in the Searching phase. Then we merge or split these candidates to remove redundancies in the Combining phase. The candidates with higher redundancy will be processed first. Finally, a reassignment step refines the result by reassigning objects and re-selecting the attributes to the candidate clusters. All phases are guided by the proposed coding scheme, and so every step guaranties decreasing coding cost, which finally leads to reaching a local minimum. The pseudocode of ROCAT is provided in Algorithm 1.

Searching Phase. We iteratively search for the best relevant Pure Subspace Cluster that reduces the coding cost most. The baseline coding cost is determined from a clustering model where all data points belong to the non-clustered area. Eq. (4) shows that large Pure Subspace Cluster will reduce the coding cost most, since the entropy of such clusters is 0. For a given searching matrix we find m large Pure Subspace Clusters, where m is the number of columns of the matrix. The pseudocode for this procedure called **FindBestPure** is depicted in Algorithm 2. The first cluster only contains attribute a with minimum entropy (see Eq. (3)) and objects with largest probability with respect

Algorithm 2 FindBestPure

Input: $Matrix$, $SubClus$
Output: C

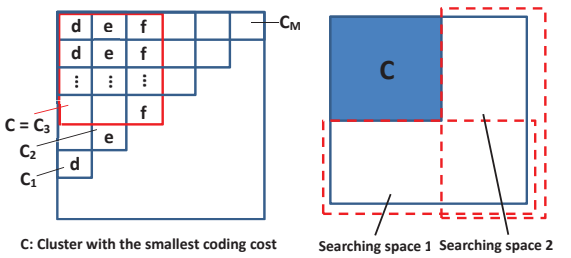
$PureClus = \emptyset$;$Att' = \emptyset$;
$Obj = Matrix.obj$;$Att = Matrix.att$;
while $PureClus.Size < |Matrix.att|$ **do**
  Find $a \in Att$ with min Entropy regarding Obj;
  $Obj' = \{o \in Obj, X_{oa} = v, |o \in Obj|$ is max$\}$;
  $Att'.Add(a)$; Form cluster C with Obj' and Att';
  $PureClus.Add(C)$; $Att.Remove(a)$; $Obj = Obj'$;
end while
$C = \{C_i \in PureClus$, Eq.(1) is min for $(SubClus \cup C_i)\}$;

return C;

to a. The second cluster is searched in the sub-matrix that contains the objects in the first cluster only. We expand the attribute set of the first cluster by the attribute that has the minimum entropy within the reduced data objects. This procedure is repeated until m Pure Subspace Clusters are found (see Figure 2a). Finally, the one that leads to minimum coding cost is returned as the best Pure Subspace Cluster. The first searching matrix is the value matrix X of data set D, in which we search for the best Pure Subspace Cluster C. If including C decreases the coding cost, we split the current searching matrix by C into two new ones (see Figure 2b) and add both to the searching matrix queue. We continue to search for best Pure Subspace Clusters until the searching queue is empty.

Combining phase. The Pure Subspace Clusters found in the Searching phase can overlap. We remove the redundancies in the Combining phase. The redundancy of each pair of clusters is modeled by their mutual information, which can be approximated by the overlapping entries between them. We firstly choose the two clusters C_i and C_j with the largest redundancy. Then we calculate the value of Eq. (1) for 4 different processing steps that are illustrated in Figure 3. We can preserve both clusters, combine the two clusters into one, preserve C_j and split C_i or preserve C_i and split C_j. Finally, we choose the step that yields the minimum coding cost. The phase is terminated when every pair of overlapped clusters has been processed.

Reassigning phase. The described Combining phase removes the redundancies by combining or splitting pairs of clusters only, thus redundancies may still exist among clusters. Besides, some objects may not be assigned to any cluster yet. Therefore we post-process the clusters in this phase to refine the result. Firstly, for each cluster C_i we



(a) Search the best C. (b) Split the search Space.

Figure 2: Searching Phase.

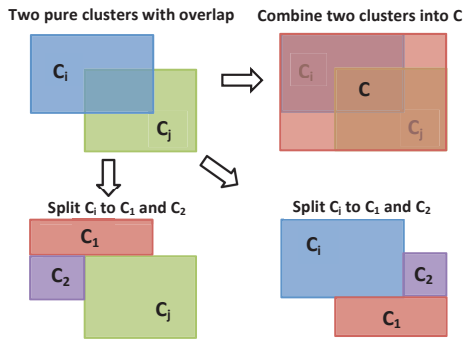


Figure 3: 4 processing candidates in Combining phase.

fix the attributes set and adjust the objects set. In detail, objects set $O \subset D.obj$ with identical value in $C_i.att$ is assigned to C_i or removed from it if this decreases the coding cost. Secondly, we fix the objects set and try to improve the subspace of each cluster. Intuitively, data objects should be compact in the subspace of C_i and sparse in the remaining attributes. Therefore, we rank the attributes according to their entropy (see Eq. (3)), since a compact attributes set leads to a lower entropy while a sparse attributes set causes a higher entropy. Finally, we compare the coding cost for the top-ranked attributes sets and keep the best one if it yields an improvement over the old attributes set. We iteratively do the two steps until no attributes or objects set changes. Those objects that still can not be assigned to any cluster are naturally regarded as outliers.

Runtime Complexity. The runtime complexity of ROCAT for a data set with N objects and M attributes can also be divided into 3 parts. In the Searching phase, we need to go through β objects α times for each dimension, where $\beta \leq N$ and $\alpha \leq M$. Suppose we find γ clusters, controlled by MDL normally $\gamma \ll N, M$. Therefore the runtime complexity in this phase is $O(\alpha \cdot \beta \cdot \gamma \cdot M)$, which is equal to $O(M^2 \cdot N)$. In the Combining phase we need to go through all pairs of clusters, so the runtime complexity in this phase is $O(\gamma^2 \cdot \kappa \cdot \iota)$, where $\kappa < N$ and $\iota < M$ are the average number of objects and attributes in each pure cluster. The runtime complexity in the Combining phase is equal to $O(M \cdot N)$. In the Reassigning phase, in each iteration for each cluster we need to go through its objects set and attributes set once. Therefore the runtime complexity in this phase is $O(i \cdot (N \cdot M))$, which is equal to $O(M \cdot N)$ since the Reassigning phase normally converges very fast. The overall runtime complexity of ROCAT therefore is $O(M^2 \cdot N)$.

4. EXPERIMENTS

In this section, we compare the performance of ROCAT to 8 methods from different areas which are related to this work. Firstly, we compare ROCAT to SUBCAD [9], CLICKS [26] and CLIQUE [2], 3 algorithms for subspace clustering on high-dimensional categorical data. CLIQUE is designed for numerical data but can be easily extended for categorical data. Moreover, we compare our work to two parameter-free algorithms for categorical data, DHCC [25] and AT-DC [7]. Due to space limitations, we do not compare to classical categorical clustering methods, i.e. K-modes [14], ROCK [13], COOLCAT [5]. These methods are not designed to find subspace clusters anyway and in addition DHCC [25] and AT-DC [7] have shown to yield better clustering models. Finally, we compare to 3 algorithms for informative itemset mining, Tiling [12], MTV [18] and Hyper+ [24], which try to find the most important itemsets. The itemset mining methods can be treated as categorical subspace clustering, since the detected itemsets can be regarded as the attributes sets of subspace clusters, while the objects that support the itemset forms the corresponding clusters. CLICKS and CLIQUE are based on the idea of itemset mining as well.

We implement ROCAT and SUBCAD in Java and use CLIQUE from the ELKI package [1]. The codes for all the other methods are provided by the authors. ROCAT, DHCC, AT-DC are parameter-free methods. SUBCAD and Tiling need the number of clusters K, where we set the true number for synthetic data and try different K for real data and output the best results. Besides, the performance of SUBCAD depends on its initialization, thus we report the average results of 10 runs. MTV is proposed as a parameter-free method, but as the execution time is too long and it allows the user to set the number of desired itemsets, we set it as for SUBCAD and Tiling. Two parameters are required for CLICKS (α and min_{sup}) and Hyper+ (false tolerant ratio f and min_{sup}) all from $[0, 1]$. We vary these parameters from 0.1 to 0.9 with a step of 0.1 for all data sets and report the best results. CLIQUE requires to pass grid size ξ and density τ as input parameters. We fix $\xi = 2 * W$ to fit categorical data, where W is the maximal number of categories. Then we vary τ from 0.1 to 0.9 with step 0.1 and report the best results. For Tiling, MTV and Hyper+, the points sets that support the detected itemsets might not cover all the points, thus we regard the rest as outliers like ROCAT.

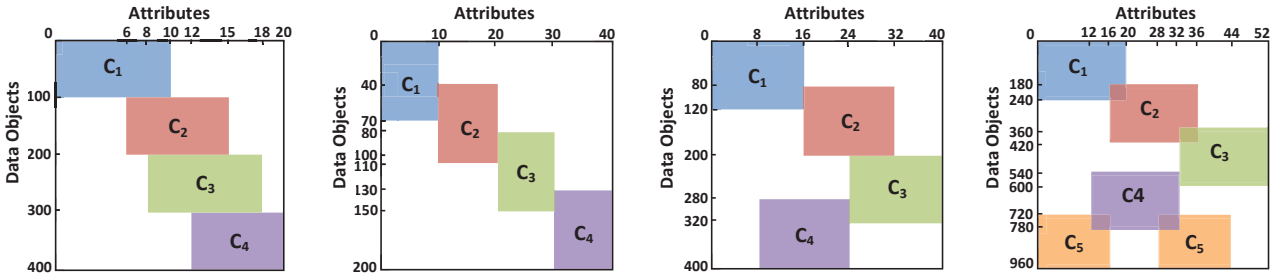
To evaluate the cluster and subspace quality, we compare pairwise Precision, Recall and F-Measure as introduced in overlapping clustering literature [4, 8] for all data sets. A pair of points sharing at least one cluster is regarded as test outcome positive in clustering results or condition positive in golden standard. Precision is calculated as $\frac{t_p}{t_p+f_p}$ and Recall is obtained by $\frac{t_p}{t_p+f_n}$, where t_p, f_p and f_n are the numbers of true positives, false positives and false negatives respectively. F-Measure is the harmonic mean of Precision and Recall. In addition, we use the confusion matrix and the cluster content to evaluate the quality of all clusters and subspaces for the used real world data.

All experiments are performed on a workstation with 2.9 GHz Intel Core i7 CPU and 8.0 GB RAM.

4.1 Synthetic Data

We generate 4 synthetic data sets with different characteristics as depicted in Figure 4. $Syn1$ contains only overlapping attributes sets, whereas $Syn2$ contains only overlapping points sets. $Syn3$ adheres both kinds of overlapping, while $Syn4$ provides a more difficult scenario. The data sets are generated by first creating Pure Subspace Clusters and then randomly choosing 10% entries of each sub-matrix and randomly changing their values. Afterwards we randomly generate values for the remaining non-clustered area. The number of categories for each attribute is randomly chosen where the average number is 4. For each scenario we generate 5 data sets and report the average performance.

Cluster Quality. Table 1 summarizes the results. Due to space limitations, the following part presents the F-Measure results only. ROCAT is the only algorithm performing very well on all the synthetic data sets with a F-Measure above



(a) Syn1. (b) Syn2. (c) Syn3. (d) Syn4.

Figure 4: Synthetic Data.

Table 1: Cluster Quality for Synthetic Data (F-Measure).

	Syn1	Syn2	Syn3	Syn4
ROCAT	**0.982**	**0.985**	**0.998**	**0.997**
SUBCAD	0.953	0.603	0.798	0.761
CLICKS	0.499	0.604	0.508	0.489
CLIQUE	0.414	0.604	0.507	0.516
DHCC	0.794	0.826	0.856	0.768
AT-DC	0.895	0.794	0.818	0.704
Tiling	0.542	0.532	-	-
MTV	0.509	0.496	0.439	0.526
Hyper+	0.565	0.634	0.565	0.491

Table 2: Subspace Quality for Synthetic Data (F-Measure).

	Syn1	Syn2	Syn3	Syn4
ROCAT	**1**	**1**	**1**	**1**
SUBCAD	0.975	0.524	0.967	0.949
CLICKS	0.742	0.375	0.375	0.528
CLIQUE	0.414	0	0	0
Tiling	0.808	0.849	-	-
MTV	0.831	0.777	0.638	0.621
Hyper+	0.565	0.469	0.744	0.853

0.982. Note that these results are obtained without requiring any input parameters from the user. Designed for categorical subspace clustering, with suitable parametrization SUBCAD performs well on data set $Syn1$ containing clusters overlapping in the attributes (F-Measure 0.95). However, the performance of SUBCAD severely degrades if clusters overlap in the objects ($Syn2$, F-Measure 0.6). CLICKS and CLIQUE perform worse than ROCAT with a F-Measure of about 0.5, since they output too many redundant clusters (thousands or tens of thousands clusters). DHCC and AT-DC perform fairly well on all the data sets with a F-Measure of about 0.8. However, DHCC and AT-DC are limited to find full-dimensional clusters and therefore do not provide any information about the subspaces in which clusters are contained. Besides, they only find partitioned clusters without any overlapping information. The three informative itemset mining methods Tiling, MTV and Hyper+ do not perform well on our synthetic data sets either and yield F-Measures of about 0.5. The results of Tiling on $Syn3$ and $Syn4$ are discarded since the running time is over 1 hour.

Subspace Quality. In contrast to traditional clustering, subspace clustering does not only aim at finding clusters but also at identifying the subspaces containing clusters with high accuracy. Table 2 shows that ROCAT is the only technique correctly identifying the subspaces in all cases with a F-measure of 1. We discard DHCC and AT-DC, since they do not support subspace clustering. SUBCAD performs well only if there is some overlap in the attributes ($Syn1$, $Syn3$ and $Syn4$), but the performance severely degrades on $Syn2$ where we only have overlap in terms of objects with an F-Measure of only 0.52. CLICKS and CLIQUE perform worst because they output too many redundant clusters. CLIQUE yields results with pairwise F-Measure values of 0 on $Syn2$, $Syn3$ and $Syn4$ because it outputs subspaces with a single attribute only. Tiling and MTV perform fairly well in terms of detecting subspaces with a F-Measure of 0.82 and 0.72 respectively. However, they only find the subsets of golden standard attributes sets. Hyper+ performs better with the more difficult scenarios $Syn3$ and $Syn4$ (F-Measure of about 0.8), but worse with the easier scenarios $Syn1$ and $Syn2$ (F-Measure of about 0.5). Since $Syn1$ and $Syn2$ are relative sparse, Hyper+ outputs more redundant clusters.

Robustness against outliers. We add different amounts of noisy objects to each synthetic data set. Particularly, we add 10% new records with random values in all attributes to $Syn1$ forming the noisy data $Syn1 - 10\%$ and analogously obtain other noisy data with different amounts. The pairwise F-Measure results are shown in Figure 5. We use the same settings for each scenario and for all the algorithms. Obviously ROCAT is extremely robust against noises. We cannot observe any decline in performance on $Syn3$ and $Syn4$. Moreover the decline is also negligible on the other two data sets yielding F-Measures above 0.96 on all examples even in the presence of 40% outliers. All the other algorithms severely degrade in performance in the presence of outliers, since they do not support the detection of noisy objects during the clustering process.

Scalability. To evaluate the scalability of ROCAT with respect to data size and dimensionality, we generate data sets using scenario 4 in Figure 4. For data size, each data set contains 52 attributes and the number of objects is varied from 10000 to 50000. For dimensionality, each data set contains 960 points and the dimensionality is varied from 50 to 200. The parameter settings are the same as for $Syn4$. Figure 6 summarizes the results. Some results are discarded if the running time is longer than 1 hour, i.e. SUBCAD and Tiling regarding data size and CLICKS and Tiling in terms of dimensionality. Figure 6 depicts that all the methods scale linearly in terms of number of objects. ROCAT performs similarly as DHCC and Hyper+, which is faster than MTV and slower than AT-DC, CLIQUE and CLICKS. With respect to dimensionality, ROCAT scales similar as DHCC, faster than SUBCAD and slower than AT-DC. CLIQUE, MTV and Hyper+ scale worst for dimensionality, where the

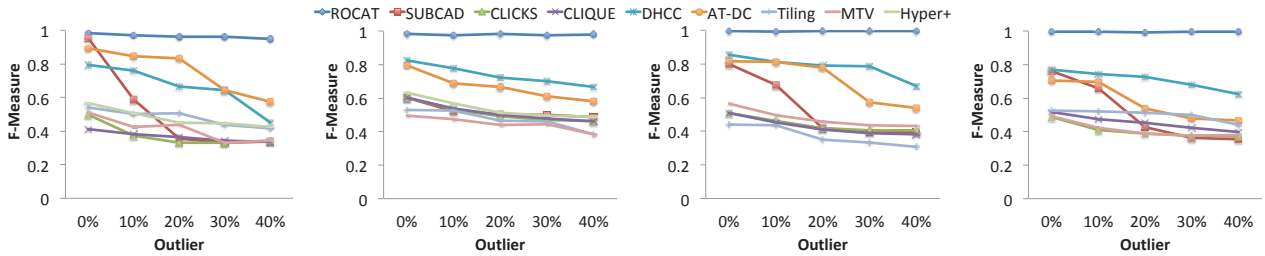


Figure 5: Robustness against outliers, *syn*1 to *syn*4 with outliers from left to right.

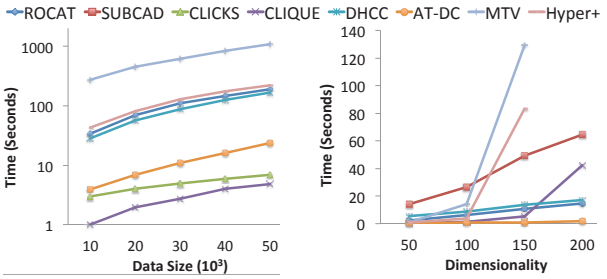


Figure 6: Scalability of ROCAT and comparisons.

running time severely increases when the dimensionality is added to 150 or 200.

4.2 Real World Data

In this section, we evaluate the performance of ROCAT and comparison methods on three real-world data sets: Congressional Votes, Mushroom and Molecular Biology (Splice-junction Gene Sequences) Data Set, which are publicly available at the UCI machine learning repository [1]. For these data sets, only non-overlapping class labels are available and there are only 2 or 3 classes. Moreover, most of the algorithms output many subspace clusters, which are normally the subsets of golden clusters. Therefore, Recall can not manifest the cluster quality anymore and we only use Precision for evaluation. We try different settings for all required parameters and choose the one with the best Precision as it is done for synthetic data sets. The results for real data sets are depicted in Table 3.

Table 3: Cluster Quality for Real Data (Precision).

	Vote	Mushroom	Splice
ROCAT	**0.812**	**0.999**	**0.861**
SUBCAD	**0.845**	0.501	0.378
CLICKS	0.525	0.508	0.343
CLIQUE	0.545	0.501	0.384
DHCC	0.793	0.766	**0.875**
AT-DC	0.521	0.612	0.497
Tiling	0.681	-	-
MTV	0.626	**0.943**	0.754
Hyper+	0.753	0.624	0.384

Congressional Votes. The data set consists of 435 instances, represented by 16 categorical attributes. There are 2 classes: democrat and republican. ROCAT and DHCC automatically output 2 clusters, while AT-DC finds 5 clusters. SUBCAD, Tiling and MTV output 2 clusters. Additionally, ROCAT, Tiling and MTV find an outlier cluster. CLICKS outputs 39 clusters, CLIQUE gives 12 clusters and Hyper+ provides 114 clusters. From Table 3 we can see that ROCAT outputs better clusters than most of the other methods with a Precision of 0.812. The confusion matrices are depicted in Table 4. Due to space limitation, we only show the results of the top 6 methods and clusters with large number of points for those with too many clusters. Clusters with high purity are highlighted in bold.

ROCAT yields two clusters with very high purity, see. Table 4a. Regarding subspace quality, the clusters found by ROCAT are more compact in the detected subspace ($Cp = 0.194$) than in the whole space ($Cp = 0.254$) and the whole data set ($Cp = 0.531$). The compactness value $Cp \in [0, 1]$ is defined in [9], and 0 means that all data values in the corresponding features are the same. Specifically, let us take a look at cluster 0 in Table 4a, which is a pure democrat cluster. ROCAT outputs 12 attributes as the subspace for this cluster. More than 95% of voters in this cluster have the same opinion in 5 of the 12 subspace attributes, they voted *yes* to *aid to nicaraguan contras*, *yes* to *adoption of the budget resolution*, *no* to *physician fee freeze*, *no* to *el salvador aid*, and *yes* to *anti satellite test ban*. Further, at least 80% of the people vote for the same in the other 5 attributes, while more than 70% of them have the same vote in the final two attributes. We get similar statistics for the other cluster. Therefore, ROCAT does find meaningful subspaces for the detected clusters. Since SUBCAD also performs very well on this data set, let us take a look at its democrat cluster as well (cluster 1 in Table 4e). The corresponding subspace consists of 3 attributes only, which represents much less information. ROCAT is able to detect higher dimensional subspace clusters due to the ability to label objects as outliers. In detail, the votes of the instances labeled as outliers are nearly averagely distributed in these 10 attributes. Therefore the properties of the outlier points are very different from those of the subspace clusters and thus it makes sense that ROCAT considers them as outliers. Tiling and MTV found outlier clusters as well. However, they can only find smaller Pure Subspace Clusters, which results in too many outliers. Some of the outliers that share similar subspaces as clusters are not detected.

Mushroom. The Mushroom data set contains 8124 records and 22 categorical attributes. Each record describes a mushroom specimen regarding 22 properties (e.g. shape, color, size) and is identified as definitely edible (4208 records) or poisonous (3916 records). ROCAT, DHCC and AT-DC automatically output 21, 10 and 6 clusters respectively. SUBCAD and MTV output 10 clusters. CLICKS outputs 260 clusters, CLIQUE gives 151 clusters and Hyper+ provides 183 clusters. Table 3 shows that ROCAT greatly outper-

[1] http://archive.ics.uci.edu/ml

Table 4: Results on Congressional Votes.

(a) ROCAT.

Cluster	Democrat	Repub.
0	**148**	**0**
1	**20**	**136**
Noise	99	32

(b) DHCC.

Cluster	Democrat	Repub.
0	49	159
1	**218**	**9**

(c) Hyper+.

Cluster	Democrat	Repub.
0	**9**	**106**
1	**107**	**2**
92	28	20

(d) MTV.

Cluster	Democrat	Repub.
0	**1**	**79**
1	**89**	**0**
Noise	177	80

(e) SUBCAD.

Cluster	Democrat	Repub.
0	**27**	**154**
1	**240**	**14**

(f) Tiling.

Cluster	Democrat	Repub.
0	**2**	**87**
1	**124**	**0**
Noise	141	81

forms the other methods with a Precision of 0.999. The confusion matrices of the top 6 methods are shown in Table 5. Not class-pure clusters are highlighted in bold.

Table 5a clearly shows that nearly all clusters detected by ROCAT are of high purity, which is much better than the other methods. Cluster 15 is the only one that contains several differently labeled records. However, DHCC, AT-DC and Hyper+ output clusters with hundreds of misclassified objects, like Cluster 0 in Table 5b, cluster 1 in Table 5d and cluster 5 in Table 5f. MTV is the second best algorithm on Mushroom data regarding Precision, because most of the clusters are Pure Subspace Clusters. However, there are still many clusters with hundreds of misclassified objects, like cluster 9 in Table 5e. CLICKS outputs many high purity clusters, however, some clusters contain 50 percent misclassified records, like cluster 210 in Table 5c. Besides, cluster 210 shows that the overlapping clusters provided by CLICKS are highly redundant. Cluster 210 contains nearly all the points and only one attribute as the subspace. In contrast ROCAT also finds overlapping clusters in the searching phase, however the redundancy is removed during the Combining and Reassigning phase. Finally, ROCAT outputs the most relevant subspace clusters without any redundancy.

In terms of subspace quality, the compactnesses Cp of the clusters found by ROCAT are 0.126 in the detected subspace, 0.171 in the whole space and 0.518 in the whole data set. Let us take Cluster 2 in Table 5a as an example. The 1296 mushrooms in this cluster are all poisonous. The subspace that this cluster exists in is composed of 14 attributes. Specifically, the cluster consists of specimen *without bruises* and *foul odor*, *free close broad gill*, with identical *shape*, *root* and *surface* of stalk, *partial white veil*, *one large ring* and the color of spore is *chocolate*. Mushrooms with these features are all poisonous. To validate the relevance of this subspace attributes, we calculate the category distribution of the remaining 8 attributes. The result indicates that the mushrooms in these attributes have different category values. For example, the *cap-shape* attribute, contains one half *bell* shaped and the other half *flat* records. Moreover, the *gill-color* attribute exhibits *buff*, *chocolate* and *green* mushrooms. Similar statistics can also be found on other clusters. Consequently, ROCAT can not only detect clusters, but can find the subspaces as well.

Splice. This data set consists of 3190 instances and 60 categorical attributes. The instances are gene sequences and attributes are the positions on the sequences. The value of each attribute is a DNA base (A, T, G, C). Splice contains class labels designating instances as either EI (767 records), IE (768 records) or Neither (1655 records). EI and IE denote that exon/intron boundaries and intron/exon boundaries can be recognized in the sequence, respectively. Neither states that there are neither EI nor IE sites.

Table 5: Results on Mushroom.

(a) ROCAT.

Cluster	Edible	Poisonous
0	1728	0
1	0	1728
2	0	1296
3	512	0
4	192	0
5	0	256
6	768	0
7	96	0
8	0	192
9	0	288
10	192	0
11	288	0
12	192	0
13	96	0
14	0	72
15	**48**	**32**
16	48	0
17	48	0
18	0	8
19	0	8
20	0	36

(b) DHCC.

Cluster	Edible	Poisonous
0	**2880**	**736**
1	**808**	**72**
2	0	1296
3	216	0
4	0	1728
5	**32**	**24**
6	**64**	**16**
7	192	0
8	0	44
9	16	0

(c) CLICKS.

Cluster	Edible	Poisonous
43	1728	0
56	0	1728
201	**2516**	**80**
210	**4016**	**3856**
230	**2016**	**8**

(d) AT-DC.

Cluster	Edible	Poisonous
0	0	192
1	**798**	**1223**
2	**62**	**1065**
3	1296	0
4	1760	0
5	0	1728

(e) MTV.

Cluster	Edible	Poisonous
0	0	1728
1	1	1296
2	1728	0
3	768	0
4	**288**	**192**
5	512	0
6	480	0
7	528	0
8	**384**	**48**
9	**496**	**224**
Noise	**192**	**524**

(f) Hyper+.

Cluster	Edible	Poisonous
1	1152	0
2	16	1296
3	0	1152
5	**855**	**353**
10	**1056**	**240**

ROCAT, DHCC and AT-DC automatically output 8, 6, and 3 clusters respectively. Besides, ROCAT identifies 1,766 points as outliers. SUBCAD and MTV output 5 clusters. CLICKS, CLIQUE and Hyper+ output 256, 241 and 399 clusters respectively. From Table 3 we can see that ROCAT outperforms most of the other methods with a Precision of 0.861. The confusion matrices of top 6 methods are shown in Table 6. Clusters with good quality regarding the number of contained points and purity are highlighted in bold.

Table 6a clearly illustrates that the clusters found by ROCAT are of very high purity. Cluster 0 and Cluster 1 contain the majority of all data points and are very pure. Cluster 0 is composed of 92% objects from class *IE* and Cluster 1 contains 97% objects from class *EI*. Besides, the outliers detected by ROCAT are mainly composed of records in the *Neither* class. DHCC and MTV also perform well on Spice. The resulting clusters are very pure as well, like clusters 1 and 4 in Table 6f and cluster 0 and the noise cluster in Table 6d. Although DHCC finds many clusters that mainly contain records of the *Neither* class, it cannot label them as outliers. On the other hand, MTV is able to find noisy cluster, but also finds clusters of lower purity compared to ROCAT. The other methods do not perform well on Splice.

The compactnesses Cp of the clusters found by ROCAT are 0.249 in the detected subspace, 0.371 in the whole space and 0.741 in the whole data set, which indicates the good quality of detected subspaces. Particularly, we choose clus-

Table 6: Results on Splice.

(a) ROCAT.

Cluster	EI	IE	Neth.
0	**45**	**703**	**16**
1	**629**	**16**	**10**
2	1	17	0
3	0	8	0
4	15	0	0
5	9	0	0
6	8	0	0
7	10	0	0
Noise	**101**	**36**	**1629**

(b) AT-DC.

Cluster	EI	IE	Neth.
0	55	78	513
1	151	644	1058
2	**561**	**46**	**84**

(c) Hyper+.

Cluster	EI	IE	Neth.
0	**374**	**8**	**3**
52	153	75	127
197	300	302	709
349	130	4	442

(d) MTV.

Cluster	EI	IE	Neth.
0	**489**	**7**	**1**
1	175	605	38
2	294	50	10
3	28	170	6
4	35	142	5
Noise	**130**	**96**	**1601**

(e) CLIQUE.

Cluster	EI	IE	Neth.
0	174	158	380
95	161	197	378
179	227	123	402
240	136	153	393

(f) DHCC.

Cluster	EI	IE	Neth.
0	15	10	419
1	**668**	**19**	**28**
2	40	3	393
3	11	0	369
4	**28**	**728**	**34**
5	5	8	412

ter 0 of ROCAT in Table 6a as an example to show the effectiveness of ROCAT on detecting subspaces. The subspace is made up of 25 out of the 60 original attributes. Among the detected 25 attributes, there are 2 positions (28 and 29) with the same values (A and G) for all sequences. Moreover 90 percent of the genes include C on position 27. Besides, there are 10 and 12 positions where more than 80 and 60 percent of all genes only take 2 different base values, respectively. On the other hand, the 4 categories $\{A, T, G, C\}$ are averagely distributed in the remaining 35 positions by nearly all the gene sequences in Cluster 0. Therefore, ROCAT outputs reasonable subspaces for the Splice data set.

CLICKS, CLIQUE and Hyper+ output overlapping clusters on Splice data set. However, there are too many clusters with a large amount of redundancies. It is hard for users to interpret such results directly. The other methods all provide partition-based results. In contrast, ROCAT finds relevant overlapping subspace clusters on Splice data set. There are 63 objects with multiple labels and 5 pairs of clusters sharing objects. Cluster 1 and 4 in Table 6a for example, share 15 records. However, they are detected in different subspaces: 6 attributes for cluster 1 and the other 52 attributes for cluster 4. Cluster 1 is very compact in the 6 detected attributes, while the 15 instances in cluster 4 are also very similar in further 52 attributes. Therefore, the overlapping clusters provide additional information over other partition-based algorithms. Further, ROCAT only provides the most relevant clusters without any redundancy, which facilitates the interpretation of the clustering results.

5. RELATED WORK AND DISCUSSION

Compared to the large body of literature on clustering numerical data only relatively few papers focus on clustering categorical data. Some prominent approaches include the basic algorithm K-modes [14] extending the famous k-means algorithm to categorical data, ROCK [13] and COOLCAT [5], to mention a few. It is often difficult to find clusters in the full dimensional space even in moderate-dimensional data sets, and a problem that is known as the *curse of dimensionality* has been extensively studied. For an comprehensive survey on clustering high-dimensional numerical data see [16]. One of the most prominent technique is CLIQUE [2]. This grid-based approach actually discretized the numerical data and therefore is also applicable to categorical data. However, it enumerates all the possible subspace clusters which produces large redundancies.

Less algorithms have been designed for categorical subspace clustering. Ganti et al. [10] proposed the categorical clustering method CACTUS, which builds a summary information from the data set first and then projects the cluster onto each attribute. It can be extended to find subspace clusters, however though introduced in the paper, it was not implemented by the authors [26]. Gan and Wu [9] proposed the categorical subspace clustering algorithm SUBCAD. They define a cost function based on the idea that data points in relevant subspaces are compact while being sparse in irrelevant ones. LIMBO [3] is a hierarchical algorithm based on an information bottleneck framework. They try to maximise the mutual information between the clusters and attribute values. A good cluster accurately predicts the attribute values associated with objects of the cluster. Although LIMBO is based on information theories it does - in contrast to ROACT - not take into account the model complexity. Furthermore, CACTUS, SUBCAD and LIMBO are all partition-based method, which cannot find overlapping clusters, and need input parameters. CLICKS [26] is a subspace algorithm which constructs a k-partite graph based on all the values of all attributes and then searches for maximum cliques. CLICKS supports overlapping clustering, however, it often includes too many redundant clusters. Besides, the input parameters are hard to determine without having deeper knowledge of the data.

Subspace clustering methods are either partition-based or produce too many redundant clusters. To solve the redundant problem, STATPC [19] and RESCU [20] are proposed to find relevant non-redundant subspace clusters in high-dimensional numerical data. However, they are not applicable for categorical data. Moreover, they need many parameters to bound the searching space.

Pattern mining is another area related to the problem of categorical subspace clustering. For instance, the frequent itemsets found by pattern mining methods could be regarded as the subspaces of clusters, while the objects that support the itemsets can be seen as clusters. Among these pattern mining algorithms, informative itemset mining, which finds the most informative itemsets or ranks the importance of the itemsets, is the most relative one. For instance, Tiling [12] defines a tile as a region in the 0/1 database where all values are 1 (Subspace Cluster), which aims at finding a tiling consisting of at most K tiles covering the largest possible area. Tiling can only find tiles without fault-tolerance, besides it needs the number of tiles as input parameter. NoisyTile [15] uses the maximum entropy distribution to measure the informativeness of a tile or tiling and it supports noisy tiles. However, it needs a fault-tolerant itemset mining algorithm, i.e. [21], to generate the candidate noisy tiles. Moreover it only gives a rank of informativeness on itemsets. Similarly Hyper+ [24] tries to find overlapped hyper-rectangles (noisy tiles) from candidates that are generated by an itemset mining method with a different cost function. KRIMP [23] and MTV [18] are designed for informative itemset mining based on compression. KRIMP needs a minimum support value as input parameter while MTV is parameter-free. However, they do not support fault-tolerant itemsets thereby performing worse than ROCAT in our experiments. The cost func-

tion of ROCAT is different and thus their searching or ranking methods cannot be directly applied. Besides, ROCAT is fully automatic while most of these algorithms need input parameters. Furthermore, ROCAT scales better than these algorithms in terms of both data size and dimensionality.

Only very few algorithms support parameter-free clustering of categorical data. Xiong et al. [25] proposes a divisive hierarchical algorithm DHCC, which iteratively splits the higher level cluster by Multiple Correspondence Analysis (MCA) and then refines the result. Cesario et al. [7] proposes a top-down algorithm AT-DC, which iteratively generates and stabilizes clusters to achieve best quality. DHCC and AT-DC are both parameter-free methods based on a top-down splitting framework, thus they can only find partitioning clusters but not overlapping clusters. Besides, DHCC and AT-DC are greatly affected by outliers, where ROCAT can handle them very well.

6. CONCLUSION

In this paper, we introduced ROCAT, an effective and efficient algorithm for detecting the most relevant overlapping subspace clusters on categorical data. Combining a compression-based view on clustering with an effective search algorithm, ROCAT identifies the most relevant subspace clusters which may overlap in terms of the assigned objects and/or the constituting attributes. The compression-based approach of ROCAT naturally avoids undesired redundancy of the result and guarantees that each detected cluster is relevant since it contributes to compress the data.

Acknowledgments

Xiao He and Jing Feng are supported by the China Scholarship Council (CSC). Claudia Plant is funded by the Helmholtz Young Investigators Groups Program.

7. REFERENCES

[1] E. Achtert, H.-P. Kriegel, and A. Zimek. Elki: A software system for evaluation of subspace clustering algorithms. In *SSDBM*, pages 580–585, 2008.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD Conference*, pages 94–105, 1998.

[3] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik. Limbo: Scalable clustering of categorical data. In *EDBT*, pages 123–146, 2004.

[4] A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R. J. Mooney. Model-based overlapping clustering. In *KDD*, pages 532–537, 2005.

[5] D. Barbará, Y. Li, and J. Couto. Coolcat: an entropy-based algorithm for categorical clustering. In *CIKM*, pages 582–589, 2002.

[6] S. Boriah, V. Chandola, and V. Kumar. Similarity measures for categorical data: A comparative evaluation. In *SDM*, pages 243–254, 2008.

[7] E. Cesario, G. Manco, and R. Ortale. Top-down parameter-free clustering of high-dimensional categorical data. *IEEE Trans. Knowl. Data Eng.*, 19(12):1607–1624, 2007.

[8] Q. Fu and A. Banerjee. Bayesian overlapping subspace clustering. In *ICDM*, pages 776–781, 2009.

[9] G. Gan and J. Wu. Subspace clustering for high dimensional categorical data. *SIGKDD Explorations*, 6(2):87–94, 2004.

[10] V. Ganti, J. Gehrke, and R. Ramakrishnan. Cactus - clustering categorical data using summaries. In *KDD*, pages 73–83, 1999.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[12] F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *Discovery Science*, pages 278–289, 2004.

[13] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *ICDE*, pages 512–521, 1999.

[14] Z. Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. In *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discoery*, 1997.

[15] K.-N. Kontonasios and T. D. Bie. An information-theoretic approach to finding informative noisy tiles in binary databases. In *SDM*, pages 153–164, 2010.

[16] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *TKDD*, 3(1), 2009.

[17] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[18] M. Mampaey, N. Tatti, and J. Vreeken. Tell me what i need to know: succinctly summarizing data with itemsets. In *KDD*, pages 573–581, 2011.

[19] G. Moise and J. Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *KDD*, pages 533–541, 2008.

[20] E. Müller, I. Assent, S. Günnemann, R. Krieger, and T. Seidl. Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data. In *ICDM*, pages 377–386, 2009.

[21] A. K. Poernomo and V. Gopalkrishnan. Towards efficient mining of proportional fault-tolerant frequent itemsets. In *KDD*, pages 697–706, 2009.

[22] J. Rissanen. *Information and Complexity in Statistical Modeling*. Springer, 2007.

[23] J. Vreeken, M. van Leeuwen, and A. Siebes. Krimp: mining itemsets that compress. *Data Min. Knowl. Discov.*, 23(1):169–214, 2011.

[24] Y. Xiang, R. Jin, D. Fuhry, and F. F. Dragan. Summarizing transactional databases with overlapped hyperrectangles. *Data Min. Knowl. Discov.*, 23(2):215–251, 2011.

[25] T. Xiong, S. Wang, A. Mayers, and E. Monga. A new mca-based divisive hierarchical algorithm for clustering categorical data. In *ICDM*, pages 1058–1063, 2009.

[26] M. J. Zaki, M. Peters, I. Assent, and T. Seidl. Clicks: an effective algorithm for mining subspace clusters in categorical datasets. In *KDD*, pages 736–742, 2005.