ISSN:0975-9646

Divide and ConquerMethod for Clustering Mixed Numerical and Categorical Data

Dileep Kumar Murala
Computer Science Engineering Department,
Nalla Malla Reddy Engineering College, Divya Nagar, A.P., India

Abstract-- Clustering is a challenging task in data mining technique. The aim of clustering is to group the similar data into number of clusters. Various clustering algorithms have been developed to group data into clusters. The main aim of cluster analysis is to assign objects into groups (clusters) in such a way that two objects from the same cluster are more similar than two objects from different clusters. Various clustering algorithms have been developed to group data into clusters in diverse domains. However, these clustering algorithms work effectively either on pure numeric data or on pure categorical data, most of them perform poorly on mixed categorical and numeric data types. In this paper we cluster the mixed numeric and categorical data set in efficient manner. In this paper, we propose a divide-and-conquer technique to solve this problem. First, the original mixed dataset is divided into two sub-datasets: the pure categorical dataset and the pure numeric dataset. Next, existing well established clustering algorithms designed for different types of datasets are employed to produce corresponding clusters. Last, the clustering results on the categorical and numeric dataset are combined as a categorical dataset, on which the categorical data clustering algorithm is used to get the final clusters.

Keywords--- clustering, novel divide-and-conquer, mixed dataset, Numerical data, and categorical data.

I. INTRODUCTION

The applications that can use clustering algorithms belong to various fields. However, most of these algorithms work with numerical data or categorical data. Nevertheless, data from real world contains both numerical and categorical attributes. In this paper we solve this problem. Clustering is considered an important tool for data mining. The goal of data clustering is aimed at dividing the data set into several groups such that objects have a high degree of similarity to each other in the same group and have a high degree of dissimilarity to the ones in different groups. Each formed group is called a cluster.

In this paper, we propose a novel divide-and-conquer technique to solve this problem. First, the original mixed dataset is divided into two sub-datasets: the pure categorical dataset and the pure numeric dataset. Next, existing well established clustering algorithms designed for different types of datasets are employed to produce Corresponding clusters. Last, the clustering results on the categorical and numeric dataset are combined as a categorical dataset, on which the categorical data clustering algorithm is employed to get the final output.

II. RELATED WORK

A. Methodology

1. Splitting of the given data set into two parts. One for numerical data and another for categorical data.
2. Applying clustering CHAMELEON algorithms for numerical data set
3. Applying clustering CACTUS algorithms for categorical data set
4. Combining the output of step 2 and step 3
5. Clustering the results using CACTUS algorithm

B. Cluster Divide and Conquer Approach for Mixed Data

Dataset with mixed data type are common in real life. Cluster Divide and Conquer is a method to combine several runs of different clustering algorithm to get a common partition of the original dataset. In the paper Divide and Conquer s technique is formulated. Existing algorithm CHAMELEON uses a graph partitioning algorithm to cluster the sparse graph data objects into a large number of relatively small sub clusters. It then uses an agglomerative hierarchical clustering algorithm to find the genuine clusters by repeatedly combining these clusters using the connectivity and closeness measures. CHAMELEON algorithm has been derived based on the observation of the weakness of two popular hierarchical clustering algorithms, CURE and ROCK.CURE and related schemes ignore information about the aggregate inter-connectivity of objects in two different clusters, they measure similarity between two clusters based on the similarity of the closest pair of the representative points belonging to different clusters. ROCK and related schemes ignore information about the closeness of two clusters while emphasizing their inter-connectivity, they only consider the aggregate inter-connectivity across the pairs of clusters and ignores the value of the stronger edges across clusters. CHAMELEON uses k-nearest neighbor graph approach to represent its objects Fig 1. This graph captures the concept of neighborhood dynamically and results in more natural clusters. The neighborhood is defined narrowly in a dense region, whereas it is defined more widely in a sparse region.

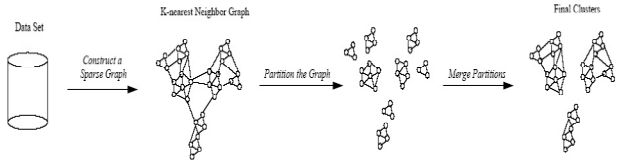

Fig 1: Overall framework of CHAMELEON Algorithm

CHAMELEON determines the similarity between each pair of clusters Ci and Cj according to their relative inter-connectivity RI (Ci; Cj), and their relative closeness RC (Ci; Cj). The relative inter-connectivity RI (Ci; Cj) between two clusters Ci and Cj is defined as the absolute inter-connectivity between Ci and Cj, normalized with respect to the internal inter-connectivity of the two clusters Ci and Cj.

Thus, the relative interconnectivity between a pair of clusters C_i and C_j is

$$RI(C_i, C_j) = \frac{|EC(C_i, C_j)|}{\frac{|EC(C_i)| + |EC(C_j)|}{2}}$$

The Edge-Cut, EC{(Ci; Cj)} (from Graph Theory) is defined to be the sum of the weight of the edges that connect the vertices in Ci to vertices in Cj. The Min-Cut bisector, EC(Ci) is the weighted sum of edges that partition the graph into two roughly equal parts. The relative closeness RC (Ci, Cj) between a pair of clusters Ci and Cj is the absolute closeness between Ci and Cj, normalized with respect to the internal closeness of the two clusters Ci and Cj

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}},$$

S EC (Ci) and S EC (Cj) are the average weights of the edges that belong in the min-cut bisector of clusters Ci and Cj, respectively, and S EC {(Ci; Cj)} is the average weight of the edges that connect vertices in Ci to vertices in Cj. CHAMELEON's hierarchical clustering algorithm selects to merge the pair of clusters for which both RI (Ci, Cj) and RC (Ci, Cj) are high.

III. METHODOLOGY

The algorithm can be divided into three different parts. The first part comprises building the k-Nearest Neighbor (k-NN) Graph from the similarity matrix. The second part consists of partitioning the graph to find initial sub-clusters, and the third part relates to merging the partitions using RI and RC values.

A. Input

The input to the first algorithm was a 26 Gene x Gene Matrix file that contained affinity values [2] (dot products of keyword relevance using z-score values) between the Genes. This matrix was read by the first C program to compute the k-NN Graph from the file. The k value was passed as a parameter to the program.

B. Converting the matrix into k-NN Graph

After obtaining the matrix, the individual genes were treated as vertices and the affinity values as the edges between the vertices. The algorithm computed the k highest affinity values from the individual genes to store as another graph consisting of k nearest vertices from each 26 set of genes.

C. Finding initial sub-clusters

CHAMELEON uses a graph partitioning algorithm to find the initial sub-clusters to partition the k nearest neighbor graph of the data set into a large number of partitions such that the edge-cut is minimized. The original algorithm used the HMetis (Hyper graph Partitioning package) library to partition the graph. HMetis is a software package for Partitioning large hyper graphs, especially those arising in circuit design. These Algorithms are fine-tuned to work on very large graphs. The 26 Gene set input was not suitable for hyper graph partitioning using HMetis. Instead, another software package library called Metis that partitions irregular graphs

Efficiently using the same edge-cut minimization procedure was used to partition the k-NN Gene graph. The input to the Metis library was a formatted graph file containing the k-NN edges and their weights. Since the program computes the multilevel k way partitioning of the graph, k value was provided as input to the program. This k denotes the initial number of sub clusters that are computed. It should be a value less than the Expected number of natural clusters which can be later discovered by the merging part of the algorithm.

The output of the program was as follows:

```
**************************************************
METIS 4.0.1 Copyright 1998, Regents of the University of Minnesota
Graph Information ---------------------------------------------------
Name: metis1.txt, #Vertices: 26, #Edges: 39, #Parts: 6
K-way Partitioning... -----------------------------------------------
6-way Edge-Cut: 535, Balance: 1.62
Timing Information --------------------------------------------------
-
I/O: 0.000
Partitioning: 0.000 (KMETIS time)
Total: 0.000
**************************************************
```

3.4. Finding Relative Interconnectivity and Relative Closeness

In order to find the RI and RC values between different clusters, the Metis program was used to get the edge-cut of different clusters efficiently. The min-cut bisector of a cluster was calculated by adding all the weights of the edges of the individual cluster. This was done under the assumption that the vertices of the cluster were all tightly connected to each other and all of them were needed to be cut to partition the graph into two roughly equal parts.

3.5. Hierarchical Merging of the clusters

Clusters are merged hierarchically if the RI and RC values are above some user specified threshold values. These threshold values control the variability of the clusters and as such have no absolute values. One can find good threshold by many experiments for a particular task. For our small Gene set, the RI threshold was chosen to be 0.1 and RC threshold was chosen to be 0.3.

IV. CACTUS

Algorithm **CACTUS (CAtegorical ClusTering Using Summaries)**, is based on the idea of the common occurrences of certain categories of different variables. If the difference in the number of occurrences for the categories *vkt* and *vlu* of the *k*-th and *l*-th variable, and the expected frequency (on the

assumption of uniform distribution in the frame of the certain categories of the remaining variables, and the assumption of the independency) is greater than a user-defined threshold, the categories are strongly connected. The algorithm has three phases: summarization, clustering and verification. During clustering, the candidates for clusters are chosen from which the final clusters are determined in the verification phase.

A. Summarization Phase

In this section, we describe the summarization phase of CACTUS. We show how to efficiently compute the interattribute and the intra-attribute summaries, and then describe the resource requirements for maintaining these summaries.

B. Clustering Phase

In this section, we describe the two-step clustering phase of CACTUS that uses the attribute summaries to compute candidate clusters in the data. In the first step, we analyze each attribute to compute all cluster-projections on it. In the second step, we synthesize, in a level-wise manner, candidate clusters on sets of attributes from the cluster-projections on individual attributes. That is, we determine candidate clusters on a pair of attributes, and then extend the pair to a set of three attributes, and so on.

C. *Validation*

We now describe a procedure to compute the set of actual clusters from the set of candidate clusters. Some of the candidate clusters may not have enough support because some of the 2 clusters that combine to form a candidate cluster may be due to different sets of tuples. To recognize such false candidates, we check if the support of each candidate cluster is greater than the required threshold. Only clusters whose support on D passes the threshold requirement are retained. After setting the supports of all candidate clusters to zero, we start scanning the dataset D. For each tuple t 2 D, we increment the support of the candidate cluster to which belongs. (Because the set of clusters correspond to disjoint interval regions, t can belong to at most one cluster.) At the end of the scan, we delete all candidate clusters whose support in the dataset D is less than the required threshold: _times the expected support of the cluster under the attribute independence assumption.

V. DCMCM ALGORITHM

In this section, we describe the DCMCM (Divide and Conquer Method for Mixed data) Algorithm framework for clustering mixed categorical and numeric data. We begin by presenting overview of the algorithm framework.

A. Overview

The steps involved in DCMCM (Divide and Conquer Method for Clustering Mixed Data) framework are described in Fig 2. First, the original mixed dataset is divided into two sub-datasets: the pure categorical dataset and the pure numeric dataset. Next, existing well established clustering algorithms designed for different types of datasets are employed to produce corresponding clusters. Finally, the clustering results on the categorical and numeric dataset are combined as a categorical dataset, on which the categorical data clustering algorithm is exploited to get the final clusters.

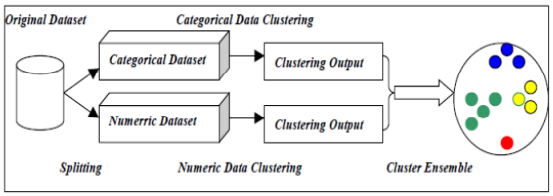

Fig 2: **Overview of DCMCM Algorithm Framework**

For this algorithm framework gets clustering output from both splitting categorical dataset and numeric dataset, therefore, it is named as DCMCM Algorithm (Cluster Divide and Conquer Based Mixed Data Clustering).

B. The Algorithm and Computation Complexity

In this section, we describe the algorithm based on DCMCM framework (DCMCM) which is described in Fig 3.

DCMCM Algorithm:

Input: The Dataset D.

Output: Each Data Object Identified.

1. Splitting the Dataset D into Categorical Dataset (CD) and Numeric Dataset (ND)

2. Clustering CD using Categorical Data Clustering Algorithm (CACTS)

3. Clustering ND using Numeric Data Algorithm (CHAMELEON)

4. Combining the outputs of above Algorithms into a Categorical Dataset (Combined CD)

5. Clustering Combined CD using CACTS Algorithm.

Fig 3: DCMCM Algorithm

The computational complexity of the DCMCM Algorithm is derived in three parts:

1) The complexity for clustering the categorical dataset,

2) The complexity for clustering the numeric dataset and

3) The complexity for clustering the combined categorical dataset

C. Scalability with Synthetic Dataset

To handle the running time, experiments were carried out with synthetic datasets. Since the complexity of the ROCK algorithm is quadratic with the number of tuples in the database, it uses sampling large datasets. The scalability of the algorithm is determined by the sample size, which makes the comparison on scalability between ROCK and CACTUS

difficult. However, we believe that, in ROCK, to preserve the quality of clustering, the sample size must be approximately set to be large enough according to the database size. With the increase of sample size, scalability of ROCK will degrade since it is quadratic with the size of sample. Thus, we compare CACTUS algorithm with Squeezer algorithm and k-modes algorithm, both of which have good scalabilities. The CACTUS algorithm scans the whole dataset twice. In the first scan summary information is collected and in the second scan clusters are produced. To make the comparison more convincing, we implemented the CACTUS in a more efficient way. In our implementation, in the first scan we only collected inter-attributes information and in the second scan nothing but clusters were labeled on the disk. Obviously, this implementation of CACTUS runs faster than the original one. For the k-modes algorithm, we set the final number of clusters to be 2 to make it run faster and save the final results onto disk. To find out how the number of tuples affects the 4 algorithms, we ran a series of experiments with increasing numbers of tuples. The datasets were generated using a data generator, in which all possible values were produced with (approximately) equal probability. We set the number of tuples to 1 million; the number of attributes to 10 and the number of attribute values for each attribute to 10. Due to sparseness of generated datasets, we set s to 1. Fig shows the scalability of Squeezer and d-Squeezer, CACTUS, and k-modes while increasing the number of tuples from 1 to 10 million. When the number of tuples goes up to 3 million, the Squeezer runs out of memory. See in Fig 4.

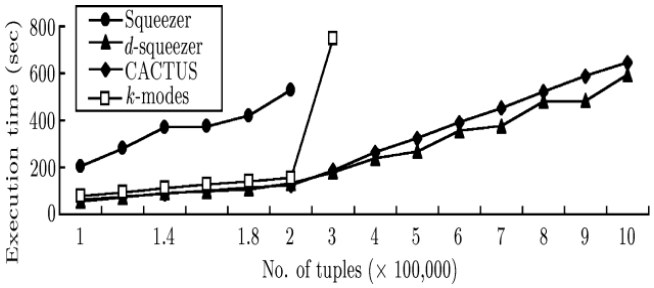

Fig 4: comparison graph

The above experimental results demonstrate the scalability of CACTUS with respect to both the size of dataset and the number of dimensions.

VI. CONCLUSIONS

In this paper, we propose a DCMCM Algorithm to solve this problem. First, the original mixed dataset is divided into two sub datasets: the pure categorical dataset and the pure numeric dataset. Next, apply CHAMELEON clustering algorithm on Numerical Dataset and apply CACTUS clustering algorithm on Categorical Dataset. Finally, the clustering results on the categorical and numeric dataset are combined as a categorical dataset, on which the designed for one type of features to handle numeric and nonnumeric feature values.

Our main contribution on this paper is to provide an algorithm framework for the mixed attributes clustering problem, on which existing clustering algorithms can be easily integrated, the capabilities of different kinds of clustering algorithms and characteristics of different types of datasets could be fully exploited.

In the future work, we will investigate integrating other alternative clustering algorithms into the algorithm framework, to get further insight into this methodology. Moreover, applying the proposed divide-and-conquer technique for detecting cluster based local outliers m large database environment with mixed type attributes will be also addressed.

REFERENCES

[1] Ming-Yi Shih*, Jar-Wen Jheng and Lien-Fu Lai,A *Two-Step Method for Clustering Mixed Categorical and Numeric Data*.

[2] 1Srinivasulu Asadi*, 2Ch. D.V. Subba Rao, 3C. Kishore and 4Shreyash Raju, Clustering the Mixed Numerical and Categorical Datasets Using Similarity Weight and Filter Method.

[3] **Saurav Sahay**, *Study and Implementation of CHAMELEON [1] algorithm for Gene Clustering*

[4] Venkatesh Ganti_ Johannes Gehrkey Raghu Ramakrishnanz, CACTUS– *Clustering Categorical Data Using Summaries*

[5] George Karypis Eui-Hong (Sam) Han Vipin Kumar, *CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling*.

[6] Zengyou He, Xiaofe i Xu, Shengchun Deng, *Clustering Mixed Numeric and Categorical Data: A Cluster Ensemble Approach**

[7] HE Zengyou, XU Xiaofei and DENG Shengchun, *Squeezer: An EÆcient Algorithm for Clustering Categorical Data*

Mr.Dileep Kumar Murala received the M.Tech degree from the Department of Computer Science Engineering, Andhra University Engineering College(A), Vishakhapatnam, in 2011 and received the B.tech degree from the Department Of Computer Science Engineering, Gudlavalleru Engineering College(GEC), JNTUniversity, Kakinada in 2009, His research interests include Data Mining and Clustering techniques.