# Clustering Categorical Data
# Using
# Data Summaries and Spectral Techniques

By

**Eman Abdu**

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2009

©2009
Eman Abdu
All Rights Reserved

This manuscript has been read and accepted for the
Graduate Faculty in Computer Science in satisfaction of the
dissertation requirement for the degree of Doctor of Philosophy.

| | |
|---|---|
| Date | Professor Bilal Khan, Ph.D.<br>Chair of Examining Committee |
| Date | Professor Theodore Brown, Ph.D.<br>Executive Officer |

Professor Spiridon Bakiras, Ph.D.

Professor Ping Ji, Ph.D.

Professor Douglas Salane, Ph.D.

Professor Nicholas Petraco, Ph.D.

Professor Gregory Conti, Ph.D.

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

**Abstract**

CLUSTERING CATEGORICAL DATA USING DATA SUMMARIES AND SPECTRAL TECHNIQUES

By

Eman Abdu

**Advisers: Professor Bilal Khan and Professor Douglas Salane**

Cluster analysis is an active area of research with applications in various fields including information retrieval, social sciences, bioinformatics, object recognition, and image segmentation (Jain et al., 1999). However, most algorithms are intended for numerical (continuous) data where proximity among data objects is naturally defined by virtue of their numerical properties. Although these algorithms can be used on categorical data, they are not designed to handle data properties typically found in this data type such as high dimensionality and lack of inherent relationships among attribute values. During the past decade, several algorithms have been designed for categorical data such as K-modes (Huang, 1998), STIRR (Gibson et al., 1998), CACTUS (Ganti et al., 1999), ROCK (Guha et al., 1999), COOLCAT (Barbara et al., 2002), LIMBO (Andritsos et al., 2004), CLICKS (Zaki et al., 2007), and others. Some of these algorithms exploit attribute relationships through data summaries such as attributes occurrence and co-occurrence frequencies while others use information entropy and links among data objects. In this thesis, we focus on using data summaries and spectral analysis to detect

clustering structure in categorical data. Spectral techniques provide a relaxed solution to the discrete clustering problem which has been shown to be NP-hard (Drineas et al., 2004). Formulating the clustering problem as a graph partitioning problem and then finding the minimum normalized cut leads to a solution based on eigenvectors of the similarity matrix (i.e. Laplacian matrix). Spectral methods have been used in various algorithms and have been shown to find non-linearly separable clusters. Equally important, spectral analysis encompasses techniques for handling high-dimensional data since input data is projected into a lower-dimensional space where all computation/comparisons can be performed. Our approach is to extend spectral techniques to data summaries which are relatively less expensive to compute than data object similarity matrix for very large data sets. Our goal is to combine the benefits of spectral analysis with the relative low cost of computing data summaries. We have developed three algorithms for clustering categorical data using data summaries. Two of them use spectral techniques. Our test results on standard data sets and synthetic data sets show that our algorithms are competitive with current spectral and non-spectral algorithms for categorical data. Our algorithms provide a solution to the categorical data clustering problem that produces quality clustering and is scalable to large data sets.

**Acknowledgments**

I would like to thank Prof. Theodore Brown, Executive Officer of the Computer Science Department, for giving me the opportunity to work on and complete my research and dissertation. I appreciate his advice and guidance throughout my doctoral studies.

I would like to express my sincere gratitude to my advisors: Prof. Bilal Khan, Chair of the Dissertation Supervisory Committee, and Prof. Douglas Salane, Director of the Center for Cybercrime Studies at John Jay College, for their support, guidance, advice, and help in pursuing and completing my research and dissertation. Prof. Khan always made himself available to address my questions and to provide thoughtful and productive discussions regarding my research. Prof. Salane provided great thought-provoking discussions that inspired me and directed my research. He was generous with his time in reviewing and providing feedback on the dissertation. He also provided me with access to the NIBRS database at John Jay, which was needed for experimental evaluations of the algorithms in this work.

I gratefully acknowledge the members of the Dissertation Supervisory Committee: Professors Spiridon Bakiras, Ping Ji, Nicholas Petraco, and Gregory Conti for their time, suggestions, and helpful feedback that led to the successful completion of this research.

I would also like to thank my father, Dr. Hani Abdu, and my mother, Dr. Afaf Abdu, for their unconditional love, continuous support, encouragement, and inspiration.

## Acknowledgments

This work is supported in part by National Science Foundation grant 0619226 and the Center for Cybercrime Studies at John Jay College.

# Table of Contents

# List of Tables

# LIST OF FIGURES

# Chapter 1 - Introduction

Clustering categorical data, i.e., data in which attribute domains consist of discrete values that are not ordered, is a fundamental problem in data analysis. Despite many advances and the vast literature in the clustering of data objects with numerical domains, clustering categorical data, where there is neither a natural distance metric nor geometric interpretation for clusters, remains a significant challenge. In addition, categorical clustering presents many of the same difficulties found in clustering numerical data, e.g., high dimensionality, large data sets and the high computational complexity associated with the discrete clustering problem. Moreover, to be effective most algorithms for clustering categorical data often require the careful choice of parameter values, which makes these algorithms difficult to use by those not thoroughly familiar with the methods.

In this thesis, we extend spectral algorithms that have proved successful in clustering numerical data to develop novel algorithms for clustering categorical data. We exploit a feature of categorical attributes not available for continuous numeric data - enumerated attribute categories. These enable us to create data set summaries based on the attribute categories occurrence and co-occurrence frequencies. By combining data summaries with spectral techniques, we develop algorithms that scale to large data sets and that are less prone to noise in data (i.e. erroneous data). In clustering numerical data, researchers have noted that spectral based techniques offer several distinct advantages: dimension reduction, noise reduction, avoidance of convergence to local minima, and the ability to account for higher order relationship effects. Moreover, spectral methods can be viewed

in the context of a graph partitioning problem and thus provide a theoretical foundation for the methods. To date, however, the use of spectral based methods for clustering categorical data largely has been ignored. By exploiting the use of data summaries, this thesis extends the benefits of spectral clustering to categorical data.

After an extensive review of non-spectral, combinatorial-based methods for clustering categorical data, we, in addition to the spectral algorithms, developed an iterative method that, like our spectral methods, is based on data summaries. The algorithm, which is somewhat similar to K-means algorithm (MacQueen, 1967), scales to large data sets. This iterative method clusters data by repeatedly refining initial clusters until a fixed point is found. This algorithm, however, offers several improvements over other algorithms in its class (K-means (MacQueen, 1967), K-modes (Huang, 1998), and K-representatives (San et al., 2004)). In this algorithm, we present a new approach for initializing cluster centers that is deterministic and not sensitive to data objects order. The algorithm is linear in terms of the number of data objects and is competitive with other clustering algorithms for categorical data in terms of clustering quality. In all of our experimental evaluations using synthetic data sets, the algorithm consistently performed better than other comparable clustering algorithms in terms of actual execution time and clustering quality.

The spectral techniques we present here extend recent work in spectral clustering (Shi and Malik, 2000; Ng et al., 2001; Zha et al., 2001; Drineas et al., 2004; Kannan et al., 2004; Ding and He, 2004) and like those methods filter out noise in data and reduce the dimensionality of a clustering data set. The spectral-based algorithms developed here combine attribute relationship and dimension reduction techniques found in methods

based on Principal Component Analysis (PCA) and Latent Semantic Indexing (LSI) that have been successful in clustering numerical data. Our algorithms differ from other spectral algorithms in that they find the spectrum of an attribute categories association matrix; most spectral-based algorithms rely on the spectrum of the data objects similarity matrix which is usually much larger than an attribute categories association matrix. Therefore, the algorithms presented are scalable to large data sets with a moderate number of attribute categories. Our particular interest in these algorithms is for clustering criminal justice data that typically has a moderate number of attributes relative to the number of data objects and the data is for the most part categorical.

This thesis contains five chapters. In Chapter 2, we review basic concepts and definitions of clustering as well as a survey of recent categorical algorithms. We present the iterative algorithm in Chapter 3. In Chapter 4, we present an extensive review of successful spectral methods and then develop our spectral methods. All experimental evaluation and comparative analysis of the algorithms are included in their respective chapters. In Chapter 5, we apply the algorithms to an application domain where the data is mostly categorical. We use crime-based data for our analysis. The data is extracted from FBI's NIBRS (National Incident Based Reporting System) which contains incident-based crime data for over 29 millions criminal incidents.

# Chapter 2 - Background and Related Work

Clustering is unsupervised learning that aims at partitioning a data set into groups of similar items. The goal is to create clusters of data objects where the within-cluster similarity is maximized (intra-cluster similarity) and the between-cluster similarity is minimized (inter-cluster similarity). One of the stages in a clustering task is selecting a clustering strategy[1] (Jain and Dubes, 1988). In this stage, a particular clustering algorithm is selected that is suitable for the data and the desired clustering type. Selecting a clustering algorithm is not an easy task and requires the consideration of several issues such as data types, data set size and dimensionality, data noise level, type or shape of expected clusters, and overall expected clustering quality. Over the past few decades, many clustering algorithms have been proposed that employ a wide range of techniques such as iterative optimization, probability distribution functions, density-based concepts, information entropy, and spectral analysis (Hartigan, 1975; Kaufman and Rousseeuw, 1990; Neville et al., 2003). In this thesis, we focus on a class of algorithms that is designed for categorical data. In this chapter, we discuss some of the well-known algorithms for categorical data as well as factors that affect the preference for a particular algorithm.

## *1 Introduction*

Clustering categorical data has gained more importance in recent years, as it becomes one of the fundamental tasks in data mining (Maimon and Rokach, 2005). Categorical data

[1] Jain and Dubes (1988) describes a clustering task as a repeating loop of seven stages where each stage depends on the outcome of the previous stage. These seven stages are: data collection, initial screening, representation, clustering tendency, clustering strategy, validation, and interpretation.

sets consist of data where the attributes only contain nominal data (discrete values that do not have any inherent order) (Hand et al., 2001). The basic goal of categorical data clustering is the same as that of clustering numerical data: form groups of data items with the objective to increase within-group similarity and inter-group dissimilarity. Clustering categorical data, however, poses a challenge not encountered in clustering numerical data. Since attribute categories are not ordered, there is no natural metric with which to measure the distance between data objects in a data set. Thus natural geometric notions of clusters as dense regions of points in a Euclidian space are not directly applicable in clustering categorical data. In addition, high dimensionality, and the large scale of contemporary data sets pose additional challenges. Finally, the computational complexity inherent in clustering numerical data also arises in clustering categorical data.

Many of the algorithms that have emerged for clustering categorical data rely on the occurrence/co-occurrence frequencies of attribute values in the data set to determine clusters of similar data objects. The basic goal is to choose a set of attribute categories that provide a summary of the data objects in a cluster. Techniques range from simple attribute value matching such as the K-modes algorithm (Huang, 1998 ) to more recent information theoretic techniques such as LIMBO (Andritsos et al., 2004). The algorithms, in our work, use the attribute categories similarity matrix and as such are scalable for large data sets with a moderate (depends on memory constraints) number of attribute categories. Typically data sets may contain millions of data objects where each data object has 10 to 50 attributes. Our algorithms are designed for clustering data sets where the number of attributes is small relative to the number of data objects in the data set.

# 2 Clustering – an Overview

Prior to discussing specific algorithms for categorical data, we provide a brief discussion of the elements that are considered when designing clustering algorithms. We provide a description of data types, proximity measures, and objective functions.

## 2.1 Data Types

The first stage in data clustering is data collection (Jain and Dubes, 1988). In this stage, a determination of what data to collect and their initial data type is made. For our discussion, each data object is represented by a vector in a d-dimensional space. Each dimension represents an attribute, a feature, or an observation. In other words, each data object has d features or attributes. The data type of an attribute refers to whether the value of the attribute is quantitative or qualitative. The value of an attribute can be classified as follows (Kantardzic, 2003):

- Quantitative. These attributes contain continuous numerical quantities where a natural order exists between items of the same data type and an arithmetically-based distance measure can be defined. Height, weight, and length are some examples.
- Qualitative. These attributes contain discrete data whose domain is finite. We refer to the items in the domain of each attribute as categories. Qualitative data are further subdivided as:
    - Nominal. Data items belonging to this group do not have any inherent order or proximity. We refer to these data as categorical data. Attributes such as color, shape, and city names are some examples of this data type.

- Ordinal. These are ordered discrete items that do not have a distance relation. Examples are ranks or ratings.

- Binary attributes are attributes that can take on only two values: 1 or 0. Depending on the context, binary attributes can be qualitative or quantitative.

## 2.2 Proximity Measures

Proximity measures are metrics that define the similarity or dissimilarity between data objects for the purpose of determining how close or related the data objects are. There are various approaches to defining proximity measures. These approaches vary from one application area to another, and depend on the data type. For most algorithms, these proximity measures are used to construct a proximity matrix that reflects the distance or similarity between the data objects. These matrices are used as input for a clustering algorithm that clusters the data according to a partitioning criterion or an objective function. For example, in some of the graph-based algorithms, the input to the algorithm is the graph adjacency matrix and the goal is to partition the graph by finding the minimum cut of the graph. In this section, we discuss some of the well-known proximity measures. Please see (Maimon and Rokach, 2005) and (Hand et al., 2001) for more discussion.

### 2.2.1 Proximity Metrics for Quantitative Numeric Attributes

Two metrics widely used to cluster numeric data are the Euclidean metric and the cosine measure. If x and y are two data objects represented as d-dimensional vectors, the Euclidean metric, denoted ||x-y||, is given by

$$|| x - y ||= \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} .$$

The cosine measure, which measures the cosine of the angle between the two vectors, is computed as

$$\frac{x^t y}{\|x\| \|y\|} .$$

where ||x|| is the norm of vector x. The value of the cosine metric is between -1 and 1. A value of 0 indicates that the two vectors are uncorrelated, a positive value indicates a positive correlation, and a negative value indicates a negative correlation.

### 2.2.2 Proximity Metrics for Qualitative Nominal Attributes

There are two methods to calculate the similarity between two data objects with nominal attributes. The first method is to convert the nominal attributes to binary attributes where each category in an attribute becomes a binary attribute that takes on the value 1 if the value is present in the data object and 0 otherwise. A proximity metric for binary attributes can then be used (see Section 2.2.4).

Alternatively, a similarity or dissimilarity metric can simply calculate the number of matches or mismatches, respectively. The number of matches or mismatches can optionally be represented as a ratio of the total number of attributes. For example, a dissimilarity measure, denoted by D(x,y), can be computed as follows:

$$D(x, y) = \frac{m}{t}$$

where m is the number of mismatches and t is the total number of attributes.

### 2.2.3 Proximity Metrics for Qualitative Ordinal Attributes

The order of the values in ordinal attribute indicates a sequence and a relation between the values of an ordinal attribute. A common approach is to standardize the values in the domain of an ordinal attribute and then apply a metric used with quantitative attributes.

### 2.2.4 Proximity Measures for Binary Attributes

A simple metric that computes the number of mismatches (hamming distance) between two vectors can be used to measure the distance between two data objects. Alternatively, the Euclidean distance or cosine measure can be used if the data objects are considered as vectors in a d-dimension space.

Another measure that is common for binary attributes is the Jaccard Coefficient. The Jaccard coefficient measures the similarity between two vectors and is defined as follows:

$$J(x, y) = \frac{A_{11}}{A_{11} + A_{10} + A_{01}}$$

Where J(x,y) is the Jaccard Similarity Coefficient. $A_{11}$, $A_{10}$, and $A_{01}$ are defined as follows:

$A_{11}$ : is the total number of attributes where both x and y have a value 1.

$A_{10}$ : is the total number of attributes in x that have the value 1.

$A_{01}$ : is the total number of attributes in y that have the value 1.

The Jaccard distance is the complement of the Jaccard coefficient and is defined as 1-J(x,y). Obviously, the Jaccard coefficient and distance do not place any importance on the case where an attribute in both x and y has a value of 0.

## 2.3 Objective Function

The aim of a clustering algorithm is to minimize or maximize an objective function. The choice of an objective function depends on the model adopted by an algorithm. For example, information-theoretic algorithms usually minimize a function that measures the entropy of the clustering (Barbara et al., 2002; Andritsos, 2004; Andritsos et al., 2004). Graph-based algorithms minimize a function that measures the Normalized Cut in a graph (Shi and Malik, 2000, Flake et al., 2004). Other algorithms are based on links between data objects and as such maximize a function based on the number of links within clusters (Guha et al., 1999). Still others simply minimize the sum of the squares of error (SSE), a popular function used in a large number of clustering algorithms (Huang, 1998; San et al., 2004).

## 2.4 Algorithms

As explained in Section 2.1, categorical attributes do not have any inherent order that would aid in the design of a numerical relationship-based similarity (dissimilarity) metric. Several algorithms have been specifically designed to cluster categorical data such as K-modes (Huang, 1998), STIRR (Gibson et al., 1998), CACTUS (Ganti et al., 1999), ROCK (Guha et al., 1999), COOLCAT (Barbara et al., 2002), LIMBO (Andritsos et al.,

2004), CLICKS (Zaki et al., 2007), and others[2]. These methods deploy various weights and proximity metrics such as simple attribute category match, attribute occurrence or co-occurrence frequencies. The K-modes algorithm is an extension of the simple and widely used K-means algorithm and naturally suffers from the same problems as the K-means algorithm, namely convergence to a local minimum, and reliance on initial cluster seeds and record order. On the other hand, CACTUS and CLICKS use attribute categories co-occurrence frequencies to discover dense regions – regions with support greater than the expected support. STIRR is an algorithm based on a non-linear dynamical system that first clusters attribute values and then clusters data objects in a post-processing step. COOLCAT and LIMBO are based on information entropy. LIMBO is a hierarchical algorithm that builds on the Information Bottleneck (IB) framework to detect the clustering structure in a data set. With the exception for STIRR, the aforementioned algorithms are described in more details in the following sections.

### 2.4.1 Notation and Definitions

The following definitions and notations will be used for the remainder of this chapter except where otherwise noted.

- For a categorical data set over attributes $A_1,\ldots,A_d$, $D_i$ denotes the domain of $A_i$ (i.e., the set of possible categories for $A_i$).
- Lower case subscripted variables, e.g., $a_i$ and $a_k$, represent categories of an attribute in a given domain.

[2] Clustering algorithms are usually classified by the type of cluster structure they produce: hierarchical or partitional. Hierarchical algorithms (agglomerative or divisive) form a tree-like cluster structure where each child node is a sub-cluster of its parent node such as single link and complete link algorithms. By contrast, partitional methods produce flat unrelated clusters such as the K-means algorithm. ROCK and LIMBO are examples of hierarchical algorithms designed for categorical data.

- The value n is the number of data objects in the categorical data set and d is the number of attributes (dimensions).
- Unless where otherwise noted, k is the number of desired clusters.
- The notation $C_j$. refers to the $j^{th}$ cluster and | $C_j$.| is the number of data objects in $C_j$.
- V denotes a set of nodes in a graph (e.g., $V=\cup_{i=1}^{d} D_i$ ) while E is the set of edges between the nodes.

### 2.4.2 K-means, K-modes, and K-representatives Algorithms

The K-means algorithm is one of the most widely used clustering algorithms due to its simplicity. The objective of the K-means algorithm (MacQueen, 1967) is to minimize the within-clusters sum of the squares of error (SSE). The K-means algorithm starts with k clusters and iteratively refines the clusters by forming a new center and re-assigning the data objects to minimize the sum of the squares of error (SSE). Mathematically, the problem is formulated as follows (Selim and Ismail, 1984; Bobrowski and Bezdek, 1991; Huang, 1998):

1) Minimize the distance between each data object and its cluster center

$$P(W,Q) = \sum_{j=1}^{k}\sum_{i=1}^{n} w_{i,j} d(X_i, Q_j)$$

2) Subject to the following constraints

$$\sum_{j=1}^{k} w_{i,j} = 1, \qquad 1 \le i \le n$$

$$w_{i,j} \in \{0,1\}, \quad 1 \le i \le n, \quad 1 \le j \le k$$

where W is an n-by-k partition matrix, $Q=\{Q_1, Q_2, \ldots, Q_k\}$ is a matrix of cluster centers, $X_i$ is a data object and d(.,.) is the square Euclidean distance between two data objects. The solution to 1 and 2 is composed of two steps that are iteratively performed until a fixed solution that satisfies 1 and 2 is found. First, Q is assigned a fixed value, and a new W is found that minimizes 1 and satisfies the constraints in 2; second W remains unchanged while a new Q is found that minimizes 1 and satisfies 2. W is calculated as follows:

$$W_{i,j} = 1 \quad \text{if } d(X_i,Q_j) \leq d(X_i,Q_t) \text{ , for } 1 \leq t \leq k$$

$$W_{i,t} = 0 \quad \text{for } t \neq j$$

Let $Q_j=(q_{j,1},\ldots,q_{j,d})$, then each element of $Q_j$ is calculated as follows

$$q_{j,s} = \frac{\sum_{i=1}^{n} w_{i,j} x_{i,s}}{\sum_{i=1}^{n} w_{i,j}}$$

for $1 \leq j \leq k$, and $1 \leq s \leq d$.

The K-means algorithm is suitable only for numeric values since the dissimilarity function measures the Euclidean distances between data objects and cluster centers, which are weighted averages of the data objects. The time complexity of the K-means algorithm is O(knj) where j is the number of iterations. To use the K-means algorithm on categorical data, Ralambondrainy(1995) proposed using binary vector representation for each data object by converting each attribute into multiple fields in a vector. Each field can have a value 1 or 0 representing either the presence or absence of an attribute category in a data object, respectively. Huang (1998) proposed the K-modes algorithm, which is an extension of the K-means clustering algorithm that is designed specifically

for categorical data. Instead of calculating the cluster centers using arithmetic means, it uses the mode of the cluster. Huang (1998) defines a mode of a set of data objects $X=\{X_1,..,X_n\}$ as a vector $Q=(q_1,\ldots,q_d\}$ that minimizes a distance function that computes the number of mismatches between the vector Q and all data objects in the set.

$$D(X,Q) = \sum_{i=1}^{n} d(X_i, Q)$$

where

$$d(X_i, Q) = \sum_{j=1}^{d} \delta(x_{i,j}, q_j)$$

Where d is the number of attributes and

$$\delta(x_{i,j},q_{j)} = 0 \quad \text{if } (x_{i,j} = q_j)$$

or

$$1 \quad \text{if } (x_{i,j} \neq q_j)$$

Alternatively, the mode of a cluster is a vector that contains the most frequent category of each attribute in the cluster.

San et al. (2004) proposed the K-representatives algorithm which is another variation of the K-means algorithm. The cluster representative is defined as follows.

$$Q_j = \{q_1, \ldots, q_d\}, \qquad \text{where } 1 \leq j \leq k$$

$$q_i = \{(a_i, f_{ai}) | \ a_i \in D_i\}, \quad \text{where } 1 \leq i \leq d$$

where $a_i$ is a category in the domain $D_i$ for the $i^{th}$ attribute, and $f_{ai}$ is the relative frequency of category $a_i$ in the $j^{th}$ cluster. Let $X_t$ be a data object vector and $f_{ai}$ be the

relative occurrence frequency of attribute $a_i$ in the $j^{th}$ cluster ($a_i$ refers to any attribute category that occurs in $X_t$), then the dissimilarity between a data object and a cluster representative is computed as follows:

$$d(X_t,Q_j) = \sum_{i=1}^{d}(1 - f_{ai}) \quad \text{where } 1 \leq j \leq k$$

A data object is assigned to the cluster whose cluster representative is least dissimilar to the data object.

When using the K-means, K-modes and K-representatives algorithms, the following important issues arise:

1) The expected number of clusters must be determined to initialize the partitions (initial clusters or cluster seeds).
2) The outcome of the algorithm is not necessarily the same over multiple executions of the algorithm (convergence to a different local minimum in each execution of the algorithm).
3) The clustering results depend on initial factors such as the order of the records and the cluster seeds.
4) The algorithms may converge to a local minimum.

### 2.4.3 ROCK

ROCK (***Ro***bust ***c***lustering using lin***k***s) (Guha et al., 1999) is an agglomerative hierarchical clustering algorithm that clusters categorical data using the concept of links between data objects instead of distances based on Euclidian measure or the Jaccard coefficient. ROCK first defines the concept of neighbors. Two data objects are

considered "neighbors" if their normalized similarity exceeds a certain user defined threshold. There are no restriction on which similarity measure to use other than it should have a value between 0 and 1. The function Link ($X_i$,$X_j$) is defined as the number of common neighbors between data objects $X_i$ and $X_j$.. ROCK seeks to maximize the sum of Link ($X_i$,$X_j$) for all data objects within a cluster and minimize the sum of that function for data objects belonging to different clusters. The objective function $E_l$ can be written as follows:

$$\sum_{l=1}^{k} n_l * \sum_{X_i, X_j \in C_l} \frac{\text{Link}(X_i, X_j)}{n_l^{1+2f(\theta)}}$$

Where $X_i$ and $X_j$ are data objects, $C_l$ is the $l^{th}$ cluster and $n_l = |C_l|$. The value $n_l^{1+2f(\theta)}$ is the expected number of links between the data objects in $C_l$. $f(\theta)$ is a function of the user-defined similarity threshold $\theta$ and is defined as $\frac{1-\theta}{1+\theta}$. ROCK defines a "goodness measure" for merging clusters $C_i$ and $C_j$ and is defined as follows:

$$\frac{\text{link}[C_i, C_j]}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}}$$

where link[$C_i$,$C_j$] defines the number of cross links between two clusters, $n_i = |C_i|$ and $n_j = |C_j|$. The value $(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}$ is the expected number of cross links between pairs of data objects in clusters $C_i$ and $C_j$.

ROCK is a two phase algorithm. The first phase creates hierarchical clusters based on links using a random sample selected from the data set. The second phase assigns the remaining data objects to the clusters created in the first phase. Each data object is

assigned to the cluster with the maximum $\frac{P_i}{(|C_i|+1)^{f(\theta)}}$ where $P_i$ is the number of neighbors in $C_i$. Let $m_m$ be the maximum number of neighbors and $m_a$ be the average number of neighbors of a data object, then the worst case time complexity for ROCK is $O(n^2 + nm_m m_a + n^2 \log n)$.

### 2.4.4 CACTUS

Ganti el al. (1999) introduces CACTUS (***Ca***tegorical ***c***lus***t***ering ***u***sing ***s***ummaries) for discovering hyper-rectangular clusters in categorical data sets which are defined as regions with support greater than a user defined threshold of the expected support for the region. Rectangular regions for categorical attributes are defined as the cross product of sets of attribute categories. CACTUS refers to these regions as interval regions. Support for a set of attribute categories is defined as the number of data objects that contain that set of attribute categories. Expected support is calculated based on the attribute independence assumption. A pair of attribute categories are said to be "strongly connected" if their support (number of data objects containing the pair of attribute categories) exceeds a threshold of the expected support for the pair. CACTUS defines a cluster as a set of attribute categories (C) where each attribute category in the cluster is strongly connected to all other attribute categories in the cluster and the support for the set (C) exceeds the expected support by a user defined threshold.

The CACTUS algorithm consists of three phases: summarization, clustering, and validation. In the summarization phases, all inter-attributes and intra-attributes summaries are calculated. Inter-attributes summarization is the process of finding all pairs of attribute categories $(a_i, a_j)$, where $a_i \in D_i$ and $a_j \in D_j$ and $j \neq i$, that are strongly

connected. Intra-attributes summarization constitutes finding all categories within an attribute that are similar. Attribute categories $a_i$ and $a_j$,where $a_i$ and $a_j$ are categories of the same attribute, are similar if $(a_i,x)$ and $(a_j,x)$ are strongly connected and x, an attribute category, is not in the same domain as $a_i$ and $a_j$.

The clustering phase synthesizes candidate clusters by extending 2-clusters to 3-clusters and so on. A k-cluster is defined as a cluster on k attributes (i.e. subspace cluster if $k \neq d$). Let $C = (C_1,\ldots,C_d)$ be a cluster on the set of attributes $\{A_1,\ldots,A_d\}$. $C_i$ is the cluster projection of C on $A_i$ and is the intersection of the cluster projections on $A_i$ of sub-clusters (2-cluster) on $A_i$ and other attributes in the cluster. In CACTUS, clusters are constructed by extending cluster projections on individual attributes to cluster projections on 2 attributes, and then to 3 attributes and so on. The computation of cluster projections on each attribute with respect to every other attribute is at least as hard as the NP-complete clique problem (Garey and Johnson, 1979). To reduce the complexity of this step, the authors of CACTUS introduce the concept of a distinguishing set. It is an assumption that there exists a positive integer *k* ("distinguishing number") where the size of a cluster projection $C_i$ on $A_i$ is larger than that number ("distinguishing number"). It follows that $C_i$ contains a set of attribute values of size *k* (or less) such that the set do not occur in any other cluster projection on $A_i$. This set is called a distinguishing set of the cluster projection $C_i$ for attribute $A_i$. A distinguishing set of size *k* (or less) is a clique of size *k* (or less) since all attribute values in the set are strongly connected (similar) to each other. The intra-attribute summaries are used to construct the distinguishing sets for each attribute. The distinguishing sets are used to construct cluster projections on individual attributes which are then extended to cluster projections on pairs of attributes.

The last phase is the validate step which calculates the support for each of the candidate clusters determined in the clustering phase. Only those candidate clusters with support exceeding a user-defined threshold of the expected support are considered final clusters.

### 2.4.5 Entropy-based Methods

COOLCAT and LIMBO (Sca***l***able ***i***nfor***m***ation ***bo***ttleneck) algorithm are based on the concept of an information-theoretic quality measure namely information entropy. Both algorithms optimize the entropy of the clustering. Entropy is a measure of the uncertainty associated with a random variable X that can take on any one of several values. Entropy is defined as follows (Shannon, 1948) :

$$H(X) = -\sum_{x \in X} p(x) \log p(x)$$

where p(x) is the probability of X taking on the value x. The lower the entropy of the random variable X, the less uncertainty one can have about an outcome. Entropy is a measure of the disorder within a cluster; therefore, the lower the entropy of a clustering, the better the quality.

#### 2.4.5.1 *COOLCAT*

COOLCAT algorithm (Barbara et al., 2002) relies on sampling of the data set to extend the algorithm to large data sets. The first phase of COOLCAT is the selection of a sample of data objects. Using this sample, COOLCAT initializes k cluster seeds where k is the number of clusters and a user-supplied parameter. In this initialization phase, COOLCAT selects the k most dissimilar data objects from the sample to be the cluster

seeds. In this selection process, COOLCAT maximizes the minimum pair wise entropy of the selected k cluster seeds. The second phase of COOLCAT is to assign the rest of the sample data objects, and then the remaining data objects in the data set, to the clusters initialized in the first phase. Since the objective is to minimize the total entropy of the clusters, the expected entropy of each data object and a cluster is computed and then the data object is assigned to the cluster with the minimum expected entropy.

#### 2.4.5.2 *LIMBO*

LIMBO (Andritsos, 2004) extends the AIB (Agglomerative Information Bottleneck) (Solnim and Tishby, 1999) algorithm by making the algorithm more scalable for large data sets. AIB is an agglomerative hierarchical algorithm that minimize the information loss (entropy-based function) resulting from merging clusters. AIB complexity is $O(n^2 d^2 \log n)$ which makes the algorithm computationally expensive for large data sets (Andritsos, 2004). LIMBO creates a summary model of the data objects that can be maintained in memory and uses the information stored in the model to merge the clusters.

LIMBO summarizes statistics about the data objects (or clusters) in a Distribution Cluster Feature (DCF). The information stored in DCF are used to merge clusters (note a cluster can also be a data object). DCF of a cluster is defined as follows:

$$\mathrm{DCF}(C_i) = (P(C_i), P(A \mid C_i))$$

Where $P(C_i)$ is the probability of cluster $C_i$ and $P(A|C_i)$ is the conditional probability of the attribute value given cluster $C_i$. DCF is stored as a B-tree where the leaves define a clustering of the data objects. LIMBO is a three-phase algorithm. The first phase is the creation (summarization) of the DCF tree. In this phase, a DCF is calculated for every

data object and then inserted into the DCF tree. The second phase is the clustering phase. The input to this phase is the number desired clusters. The authors of LIMBO recommend using the AIB algorithm to merge the DCFs but they note that any algorithm can also be used in this phase. The last phase associates each data object with the DCF closest to it. Note that the model is based on the work by Tishby et al.(1999) where the information loss can be calculated using the probability of the cluster, the conditional probability of the attributes given a cluster, and Jensen-Shannon(JS) divergence.

To illustrate the time complexity of LIMBO, let B be the branching factor of the DCF tree, h be the height of the DCF tree, U be the number non-leaf nodes, and L be the number of DCF entries at the leaves of the tree. The time complexity of the first phase is $O(nhdB + dUB^2)$, for the second phase is $O(L^2 m^2 \log L)$ if AIB is used, and for the third phase is $O(kmn)$.

### 2.4.6 CLICKS

(Zaki et al., 2007) discuss three issues unique to categorical data: lack of pre-defined order of the domain values of categorical attributes, high dimensionality, and subspace clusters. (Zaki et al., 2007) point out that current algorithms usually address some of these challenges and none deals with them all. (Zaki et al., 2007) introduces CLICKS, a k-partite graph search-based algorithm designed for categorical data that is scalable and handles subspace clusters.

In CLICKS, the clustering problem is defined as finding the maximal, dense, and strongly connected k-partite cliques in an undirected k-partite graph. The data set is summarized as a k-partite graph where k corresponds to the number of attributes in the

data set. Each node in the graph is a category of an attribute. An edge between nodes $V_i$ and $V_j$ exists if the support of $V_i$, and $V_j$ exceeds a user-defined threshold of the expected support for these two nodes. In other words, an edge between two nodes (attribute categories) can exist only if they co-occur together in data objects and the number of their co-occurrence exceeds a pre-defined threshold based on expected support. As defined for CACTUS, support is the number of data objects that contain a given set of attribute categories. Expected support is calculated based on assumed probability distribution of the attributes; the authors of CLICKS assume attribute independence but they note that weights and preferences for certain attribute categories can easily be incorporated in computing the expected support for any given set of attributes.

The CLICKS algorithm is a 3-step process. The first step is for pre-processing the categorical data set. In this step, the k-partite graph is created based on the support among the nodes. Additionally, each attribute is ranked based on the notion of connectivity (a function of the neighbors of an attribute category and number of categories in the attribute domain). The number of neighbors of a node is defined as follows:

$$N(V_j) = \{ V_k \in V: (V_j,V_k) \in E\}$$

The connectivity of a node is defined as the number of neighbors it has plus the number of remaining categories in its domain. If a node does not have any neighbors, then its connectivity is zero. Since, two categories of the same attribute cannot appear in the list of neighbors for each other (i.e. no two categories of the same attribute can appear in the same data object), the authors of CLICKS have considered categories within an attribute to be implicitly connected. The connectivity of each node (attribute

category) is used in ranking the nodes for the clique enumeration process. As per the authors, the attribute ranking is only needed for efficient mining of the graph cliques. The second step of CLICKS is the enumeration of the k-partite cliques (maximal and strongly connected). The third and last step is the post-processing step (validation step). In this step, each maximal clique found is verified to have support above a user-predefined threshold of the expected support for the given clique. This step is necessary since a set of three strongly connected attributes may be due to different sets of data objects. For completeness, the authors propose a "selective vertical expansion" process to find all sub-cliques that may have been missed in the pruning process (step 2: cliques enumeration). The author also extend the algorithm to merge clusters based their common coverage (number of data objects common to the clusters). This optional step of CLICKS is a merge step aimed at merging overlapping cliques into large cliques to reduce the final number of cliques.

A maximal clique is a complete sub-graph that is not part of any other complete sub-graph. Enumerating the maximal cliques of a graph is a NP-hard problem (Garey and Johnson, 1979). (Zaki et al.,2007) use an algorithm similar to BK algorithm (Bron and Kerbosch, 1973) for finding maximal cliques but customized for k-partite graphs. (Moon and Moser, 1965) show that a graph with n nodes can have as many as $3^{n/3}$ maximal cliques which gives a worst case scenario time complexity of $O(3^{n/3})$ (Tomita et al., 2006).

According to Zaki et al. (2007), on average, CLICKS is O(dm$\mathcal{L}$) for the clique mining step, O($2^{l}\mathcal{L}$) for the selective expansion step, and O(d$\mathcal{L}\mathcal{F}$) for the merge step where d is the number of attributes, m is the number of categories in each attribute

domain, $\mathcal{L}$ is the number of maximal cliques, $l$ is the length of the longest clique, n is the number of records in the data set, and $\mathcal{F}$ is the number of maximal cliques a data object belongs to. Overall, CLICKS can be an exponential time algorithm in the worst case.

### 2.4.7 Spectral-based Algorithms

In this section, we present a brief overview of spectral-based algorithm. We defer a more detail discussion to Chapter 4 where we present two novel algorithms based on spectral techniques. Spectral techniques provide a relaxed solution to the discrete clustering problem which has been shown to be NP-hard (Drineas et al., 2004). Spectral-based clustering algorithms solve the clustering problem by finding the first few eigenvectors of a matrix computed based on the similarity between the data objects. These eigenvectors form a relaxed (continuous) solution for the cluster membership indicator vectors. A key feature of these methods is that they are not vulnerable to the local optimum problem as in the K-means like algorithms.

### 2.4.8 Discussion

The K-modes and K-representatives algorithms extend the K-means algorithm to categorical data but they still suffer from the same issues as the K-means algorithm namely conversion to a local minimum and sensitivity to initial conditions. Yet, both algorithms are linear and are efficient for large data sets.

ROCK is a hierarchical agglomerative algorithm based on links between data objects. CLICKS, LIMBO, and COOLCAT have been shown to outperform ROCK (Ganti et al., 1999; Andritsos et al., 2004; Zaki et al., 2007). The time complexity of ROCK makes the algorithm unsuitable for large data sets. The authors of COOLCAT

compared their algorithm to ROCK and showed a small advantage in terms of clustering quality but they assert that the user-defined threshold used in ROCK is extremely difficult to tune (Barbara et al., 2002). The authors of LIMBO, also, observe that the performance of ROCK is very sensitive to the user-defined threshold.

COOLCAT and LIMBO are based on the information entropy. LIMBO is an agglomerative hierarchical algorithm whereas COOLCAT is not. COOLCAT uses sampling to allow the algorithm to scale to large data sets while LIMBO relies on a compact summary model. COOLCAT clustering quality may vary depending on the sample size and data objects order; consequently, different runs of the algorithm may produce different clusterings. (Andritsos, 2004; Andritsos et al., 2004) compared COOLCAT to LIMBO, and showed that LIMBO is less sensitive to the order of data objects than COOLCAT using data sets from UCI Machine Learning Repository (Asuncion and Newman, 2007). The authors of LIMBO presented comparative analysis with COOLCAT and showed that LIMBO outperforms COOLCAT in terms of parameter stability and clustering quality. The authors of LIMBO also showed that their algorithm outperforms STIRR and ROCK in terms of clustering quality.

CACTUS and CLICKS have similar definitions of what a cluster is but use different methodology to find the clusters. CLICKS casts the problem of clustering as a problem of finding the maximal cliques in a k-partite graph of attribute categories where each maximal clique corresponds to a candidate cluster. CACTUS finds the clusters by merging cluster projections on individual attributes. Ganti et al. (1999) showed that CACTUS outperforms STIRR on synthetic data sets in terms of speed and scalability and showed that STIRR fails to find clusters otherwise found by CACTUS. Zaki et al.

(2007) present a comparative study of CLICKS, CACTUS, ROCK and STIRR where they show that CLICKS outperformed all of these algorithms in terms of clustering quality and scalability on synthetic data sets.

CLICKS mines subspace clusters (a feature that is not common in clustering algorithms). CACTUS can only mine subspace clusters that occur in the same order that the attribute projections are processed.

Regarding time complexity, the K-modes, K-representatives and COOLCAT algorithms are linear in terms of the number of data objects. ROCK is quadratic in terms of the data objects. In worst case, CLICKS can be exponential in terms of the number attribute categories (Zaki et al., 2007).

The aforementioned algorithms have been designed specifically for categorical data to remedy the fact that the categorical attributes lack pre-defined order among the categories of any single attribute. Some of these algorithms run in linear time but may produce less than optimal quality clusters; others may produce quality clustering but may run in exponential time in worst case. Also, all the aforementioned algorithms require the careful choice of the input parameters to deliver the optimal results.

## 3 Remarks

In this chapter, we discussed current popular algorithms for categorical data and compared the algorithms in terms of their time complexity, input parameters, and scalability. Although, some of the aforementioned algorithms provide high quality clustering, they may be computationally expensive for large data sets. For example, CLICKS and CACTUS are based on heuristics for solving the enumeration of maximal cliques and computing cluster projections of 2-clusters, respectively, which are NP-hard

problems (Garey and Johnson, 1979). On the other hand, ROCK is quadratic in terms of the number of data objects when sampling is not used. In our opinion, each of the algorithms presented in this chapter has some disadvantages that may affect its outcome or limit its applicability to large data sets. For instance, the quality of the clustering is often dependent on the careful choice of input thresholds which are usually not easy or intuitive to determine. CLICKS and CACTUS require as input a threshold rate for defining minimum support for sets of attributes categories. Likewise, not only do COOLCAT and ROCK require user-defined thresholds, but they also rely on sampling which may result in lower quality clustering if the sample mostly contains outliers. Also, ROCK, COOLCAT, K-modes, K-representatives, and LIMBO require as input the number of desired clusters, which is not always readily available or easy to determine.

In this thesis, we will present three novel algorithms that provide clustering results that are competitive with current clustering algorithms for categorical data in terms of clustering quality and time complexity. In Chapter 3, we present a novel algorithm that uses an iterative process to find the clusters with few or no input parameters. In Chapter 4, we present two novel spectral-based algorithms that combine data summaries and spectral techniques to produce quality clusters; the algorithms are scalable to large data sets.

# Chapter 3 - CATS - A K-means-like Algorithm with a Deterministic Method for Cluster Initialization

In this chapter, we introduce a partitioning algorithm for clustering categorical data based on data summaries. The algorithm is a K-means-like algorithm that builds on the concept of cluster representatives presented in San et al. (2004). We will use the acronym CATS to refer to the algorithm (***C***lustering c***a***tegorical data using a***t***tribute ***s***ummaries).

## *1 Introduction*

CATS introduces a deterministic method for initializing clusters' seeds to overcome several problems associated with K-means-like algorithms such as different clustering results produced from different executions of the algorithm and the need to determine the number of clusters beforehand. Based on experimental evaluation on synthetic and standard test data sets (Asuncion and Newman, 2007), CATS' clustering quality is competitive with the K-representatives algorithm as well as other algorithms designed for categorical data.

## *2 Background and Related Work*

As discussed in Chapter 2, the K-means algorithm (MacQueen, 1967) is a greedy algorithm that iteratively refines the current partitioning of a data set by minimizing an objective function such as the sum of the squares of error (SSE). The K-modes (Huang, 1998) and K-representatives (San et al., 2004) algorithms extend the K-means algorithm by defining a new objective function and changing the method of defining a cluster representative to make the algorithm more suitable for categorical data. The input

parameters to all K-means-like algorithms are the number of desired clusters as well as initial clusters or cluster seeds. Over the years, several methods have been proposed for initializing the clusters for the K-means algorithm. Random partitions and random seeds methods remain the most popular initialization techniques used in the K-means algorithm. Random partitions method basically separates a data set into k partitions by randomly assigning each data object to one of the k partitions and then calculates a new cluster center for each partition. On the other hand, random seeds method randomly selects k data objects to be k cluster seeds (representatives). Other methods include a method proposed by MacQueen(1967) which enhances random seeds method by assigning data objects (in order) to the closest seed and re-computing a new cluster seed after each assignment. As for categorical data, these methods are modified to work with data sets of categorical attributes. Recently, a method specific to categorical data has been proposed by Khan and Kent (2007). Khan and Kent (2007) introduces an initialization method that uses Evidence Accumulation, which compute cluster modes for a final execution of the algorithm based on data accumulated from multiple prior K-modes clustering results for the data set; however, this initialization process may not be practical for large data sets.

It is well known that the performance and the quality of the clustering produced by K-means-like algorithms depend on the cluster initialization method, data object order, and number of clusters specified. More importantly, the output of K-means-like algorithms will vary between different executions of the algorithm since almost all initialization methods are non-deterministic and the K-means algorithm may converge to a different local minimum in each execution if the initial conditions are different (cluster

seeds or data objects order). In general, multiple executions of K-means-like algorithms are needed to determine the partitioning of the data set that is optimum given an objective function and a number of desired clusters.

# 3 CATS

In this chapter, we introduce a novel algorithm that uses a deterministic method for initializing the cluster seeds (representatives) using the attribute categories similarity matrix. Our initialization method removes the burden of trying to predict the number of clusters in a data set (a key weakness in many clustering algorithm including K-means, K-modes and K-representatives algorithms). Our notion of a cluster representative is based on the work of San et al. (2004). San et al. (2004) define a "cluster representative" as a vector of attribute categories and the relative occurrence frequency of each attribute category in the cluster. In this chapter, we will use the term "cluster representative" to refer to a vector that represents a cluster even though our definition of a cluster representative is slightly different (see Section 3.3) from that specified in San et al. (2004).

## 3.1 Data Representation, Definitions, and Notation

In handling categorical data, we follow a common approach that converts categorical data objects into binary indicator vectors (Ralambondrainy, 1995; Cheng et al., 2006). Each indicator corresponds to one category in the domain of each attribute[3]. Algorithms such as LIMBO, CLICKS and CACTUS do not model the data objects as binary vectors but

[3] Most practical data sets in data mining contain categorical attributes with few values. For attributes with large domains, low-frequency attribute categories can often be eliminated without impacting the cluster analysis.

the underlying algorithm methodology and computations are done on the attribute category level, effectively expanding each attribute into its categories. The following definitions will be used for the remainder of the chapter.

- Let $A_1,\ldots,A_d$ be a set of categorical attributes with domains $D_1,\ldots,D_d$, respectively. Each data object denoted $X^i$ is a vector in the d-dimensional space $D_1 \; x \; \ldots \; x \; D_d$. The matrix X is a d-by-n matrix of the data objects. The data objects form the columns of X. $X^i$ refers to the $i^{th}$ column in the matrix and $X_j$ refers to the $j^{th}$ row of the matrix.
- The value $a_i$ represents an attribute category.
- The m-by-n binary matrix Y represents the data objects after each attribute is expanded into a binary representation. Here $m = |D_1| + \ldots + |D_d|$ and $|..|$ denotes the number of categories in all domains. Thus each data object is represented by an m-dimensional binary vector where each component corresponds to an attribute category ($a_i$). The value of the component is 1 if the corresponding attribute category is present in the data object; otherwise, the value is 0. Again the columns of matrix Y correspond to the data objects and the rows correspond to attribute categories. $Y^i$ refers to the $i^{th}$ column of the matrix and $Y_j$ refers to a $j^{th}$ row of the matrix.
- Let Q be a k-by-m matrix of cluster representatives (see Section 3.3) where the cluster representatives are the rows of Q and k is the number of clusters. The vector $Q_i =(q_1,\ldots,q_m)$ refers to the $i^{th}$ row of matrix Q.
- If $C_i$ is a cluster, then $Q_i$ is a vector that represents $C_i$.

- Let $f_i$ and $f_j$ be the occurrence frequency of the $i^{th}$ and $j^{th}$ attribute categories; and let $f_{i,j}$ be the co-occurrence frequency of the $i^{th}$ and $j^{th}$ attribute categories. To reference the occurrence frequency of a specific category, we will use the notation $f_{ai}$.
- Let F be a k-by-m matrix of frequency vectors where each row contains the frequency of all attribute categories in a cluster. For example, $F_i$ contains the occurrence frequency of all attribute categories in the $i^{th}$ cluster.
- The m-by-m matrix T is the attribute categories similarity matrix.
- The vector e is a constant row vector whose components are all 1.

Where it is evident from the context, we will use the term attributes to refer to attribute categories.

## 3.2 Cluster Initialization

Good cluster seeds are those that are evenly distributed across a data set. Our approach is to find candidate cluster seeds that are distributed over a data set based on data summaries. A brute force search strategy to clustering is to partition a data set based on the categories in each attribute. Each attribute partitions a data set into disjoint clusters where each cluster contains the data objects that share the same value (category) of the attribute. Each cluster can then be partitioned by another attribute and the process would be repeated until all attributes have been processed. This partitioning strategy would create multiple tree-like structures where the path from the root to each leaf represents a set of attribute categories that corresponds to a cluster. The goal is to find sets of attribute categories that occur together in the data set more often than all other sets (i.e., a large number of data objects contain the specified set of attribute categories). However,

instead of processing every possible combination of the attribute categories, some parameters can be defined such as the order of processing the attributes, and the threshold for a cluster size to make the search more tractable. In addition to high time and space complexity of this search strategy, finding the clusters is dependent on the processing order of the attributes. Consequently, the significance of each of the attributes in forming the clusters would need to be known beforehand; otherwise, certain clusters may not be discovered. However, this brute force search strategy illustrates that the clusters can be found be searching for strongly related-attribute categories (attribute categories that commonly co-occur in the data). Algorithms such as CLICKS and CACTUS are based on the concept of strongly-connected (similar) attribute categories. Our approach is not to partition the data set on each attribute but to create cluster representatives for clusters that are defined for each attribute category (or attribute categories that are statistically significant). We initialize the cluster representatives to reflect the relationship between each pair of attribute categories. We create m cluster representatives (m is the total number of attribute categories). Clusters that corresponds to categories of the same attribute are disjoints but clusters on different attribute categories may overlap.

We define the cluster representative ($Q_i$) in terms of the occurrence frequency of each attribute category in a cluster ($C_i$) that is created on any one dimension. $C_i$ is a cluster of the data objects that contain the $i^{th}$ attribute category.

$$q_{i,j} = \{f_{a_j} \mid a_j \in D_j\}$$

The attribute categories co-occurrence frequency matrix contains the cluster representatives as defined above. Each row of the attribute categories co-occurrence frequency matrix contains the co-occurrence frequency of the attribute category

associated with that row and all other attribute categories. Thus the attribute categories co-occurrence frequency matrix contains the cluster representatives according to our definition. In other words, each row of the attribute categories co-occurrence frequency matrix can be viewed as 1) the cluster representative for a cluster defined on the attribute category associated with that row (i.e., a cluster of all data objects containing the attribute category corresponding to that row of the matrix); 2) an m-vector that provides values for the degree of association between an attribute category and all other attribute categories.

Instead of using absolute co-occurrence frequency, we use normalized co-occurrence frequency obtained by the cosine similarity between the attribute categories ($Y_i$). In other words, each co-occurrence frequency of a pair of attribute categories is relative to the product of the square root of the occurrence frequency of each attribute category. Each cell in the attribute categories similarity matrix is defined as follows:

**Equation 3-1**

$$\frac{f_{i,j}}{\sqrt{f_i}\sqrt{f_j}}$$

Alternatively,

$$q_{i,j} = \{\frac{f_{a_i,a_j}}{\sqrt{f_{a_i}}\sqrt{f_{a_j}}} \mid a_i \in D_i, a_j \in D_j\}$$

The notion is that if we have knowledge as to which set of attribute categories form a cluster, then we would use the brute force method, mentioned previously, to partition the data set on these categories. In the absence of such information, we assume that all attribute categories are critical and as such we create a cluster representative for each category (dimension). Note that the definition of a cluster representative (cosine similarity) ensures that attribute categories that occur together will have high weights

(similarity). For example, if categories $a_1$, $a_2$, and $a_3$ (where $a_1 \in D_1$, $a_2 \in D_2$, $a_3 \in D_3$, and $D_1 \neq D_2$, $D_2 \neq D_3$, $D_1 \neq D_3$) co-occur frequently together, then we would expect that the cosine similarity of $(a_1,a_2)$, $(a_1,a_3)$, $(a_2,a_3)$ to be relatively high in the attributes similarity matrix's rows associated with $a_1$, $a_2$, and $a_3$. Note that the attribute categories cosine similarity matrix is symmetric; that is the cosine similarity of $(a_1,a_2)$ is the same as cosine similarity of $(a_2,a_1)$.

## 3.3 Cluster Representatives

Our definition of a cluster representative will depend on when it is used in the algorithm. Equation 3-1 defines the initial cluster representatives (seeds or candidate cluster representatives) in terms of the attribute categories co-occurrence frequency and the frequency of each of the attribute categories in the data set. Equation 3-1 also defines each cell in the attribute categories similarity matrix (cosine measure). The second definition (Equation 3-2) of a cluster representative is used subsequently in the algorithm and is defined as follows:

**Equation 3-2**

$$Q_j = \frac{F_j}{\| F_j \|}$$

## 3.4 A simple Illustration

### 3.4.1 Example 1

We work through a small example to illustrate the approach. The tables presented in Table 3-1 and Table 3-2 show a binary data matrix and its attribute co-occurrence frequency matrix. The data matrix contains the data for 5 data objects (columns) and 6

attributes (rows). Note that the first three data elements form one cluster while data objects 4 and 5 form another cluster (Table 3-1). The attributes cosine similarity is presented in Table 3-3. Table 3-4 shows the inner product between the attributes similarity and the data objects matrices. Each data object is assigned the row (cluster) that has the highest inner product (weight) for that data object. Based on the data presented in Table 3-4, data objects 1, 2, and 3 have the highest weight with the first row (or third row) and as such can be clustered together. Data objects 4 and 5 form the second cluster (second or fourth row). Therefore, the clustering accuracy in this example is 100%.

| | | Data elements | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| Attribute Values | A1 | 1 | 1 | 1 | 0 | 0 |
| | A2 | 0 | 0 | 0 | 1 | 1 |
| | B1 | 1 | 1 | 1 | 0 | 0 |
| | B2 | 0 | 0 | 0 | 1 | 1 |
| | C1 | 1 | 0 | 1 | 1 | 0 |
| | C2 | 0 | 1 | 0 | 0 | 1 |

**Table 3-1 – Example 1 - Data objects matrix**

| | | Attribute Values | | | | | |
|---|---|---|---|---|---|---|---|
| | | A1 | A2 | B1 | B2 | C1 | C2 |
| Attribute Values | A1 | 3 | 0 | 3 | 0 | 2 | 1 |
| | A2 | 0 | 2 | 0 | 2 | 1 | 1 |
| | B1 | 3 | 0 | 3 | 0 | 2 | 1 |
| | B2 | 0 | 2 | 0 | 2 | 1 | 1 |
| | C1 | 2 | 1 | 2 | 1 | 3 | 0 |
| | C2 | 1 | 1 | 1 | 0 | 0 | 2 |

**Table 3-2 – Example 1 – Attributes co-occurrence frequency matrix**

| | | Attribute Values | | | | | |
|---|---|---|---|---|---|---|---|
| | | A1 | A2 | B1 | B2 | C1 | C2 |
| Attribute Values | A1 | 1.000 | 0.000 | 1.000 | 0.000 | 0.667 | 0.408 |
| | A2 | 0.000 | 1.000 | 0.000 | 1.000 | 0.408 | 0.500 |
| | B1 | 1.000 | 0.000 | 1.000 | 0.000 | 0.667 | 0.408 |
| | B2 | 0.000 | 1.000 | 0.000 | 1.000 | 0.408 | 0.500 |
| | C1 | 0.667 | 0.408 | 0.667 | 0.408 | 1.000 | 0.000 |
| | C2 | 0.408 | 0.500 | 0.408 | 0.500 | 0.000 | 1.000 |

**Table 3-3 - Example 1 - Attributes similarity matrix (cosine measure)**

| | | Documents | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| Attribute Values | A1 | 2.667 | 2.408 | 2.667 | 0.667 | 0.408 |
| | A2 | 0.408 | 0.5 | 0.408 | 2.408 | 2.5 |
| | B1 | 2.667 | 2.408 | 2.667 | 0.667 | 0.408 |
| | B2 | 0.408 | 0.5 | 0.408 | 2.408 | 2.5 |
| | C1 | 2.333 | 1.333 | 2.333 | 1.817 | 0.817 |
| | C2 | 0.817 | 1.817 | 0.817 | 1 | 2 |

**Table 3-4 - Example 1 –Attributes similarity matrix and data objects inner product**

## 3.5 An Object Complement

Our approach to initializing the cluster representatives is based on the co-occurrence frequencies of the attributes. The approach is to represent a cluster on each dimension in the data set (a cluster of data objects that contain an attribute category (dimension)) with a vector that contains the normalized occurrence frequency of each attribute category in this cluster. We call this vector a cluster representative. Note that the cluster representative contains the co-occurrence frequency between the attribute category represented by this cluster and all other attribute categories. It is possible that several attributes may have high co-occurrence frequency or an attribute may occur in all data objects. The cluster representative for these attributes will have high co-occurrence

frequencies and as such may act as a "catch all" cluster. To avoid falsely clustering the data objects with these "catch all" clusters, we modify our objective function to assign a data object to a cluster candidate that is

*most similar to the data object and least similar to its complement.*

We define a data object's complement as a vector that is orthogonal to the data object. (We will refer to a data object's complement as complement, for short, in the rest of this chapter). A data object and its complement will have a cosine similarity of zero. A complement will have 1's where there are zeros in the data object binary vector and 0's otherwise. To achieve our objective, a data object will be assigned to a candidate cluster where the difference between the similarities of the data object and its complement to the candidate cluster is maximized.

### 3.5.1 Example 2

We now illustrate the use of the complement with a small example. Table 3-5 presents the binary data for 6 data objects with 6 attributes. In this example, a possible grouping is to have data objects 1-3 in one cluster and data objects 4-6 in another cluster. Table 3-6 contains the data objects' complements.

Table 3-7 shows the inner product between the attribute similarity and data objects matrices and Table 3-9 shows the difference between the inner product of the attribute similarity matrix & the data objects and the attribute similarity matrix & the complements. Based on the data in

Table 3-7, all data objects would be erroneously grouped into one cluster. However, when the candidate cluster assignments is also based on the similarity to the complement,

the data objects would be clustered properly as shown in Table 3-9. Furthermore, using the inner product of the attribute similarity and data objects' complements matrices will not result in the correct clustering (Table 3-8). Note that certain data objects may have equal weights with more than one candidate cluster. This reflects the fact that some attributes may have the same co-occurrence frequency distribution as other attributes and as such they form alternative candidate clusters. For example, the sixth data object can be clustered with the fourth and fifth data objects or form its own cluster.

| Attribute Values | Data Objects | | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| | a1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | a2 | 0 | 0 | 0 | 0 | 0 | 0 |
| | b1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | b2 | 0 | 0 | 0 | 1 | 1 | 1 |
| | c1 | 1 | 0 | 1 | 0 | 0 | 1 |
| | c2 | 0 | 1 | 0 | 1 | 1 | 0 |

**Table 3-5 - Example 2 - Data objects matrix**

| Attribute Values | Data Objects | | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| | a1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | a2 | 1 | 1 | 1 | 1 | 1 | 1 |
| | b1 | 0 | 0 | 0 | 1 | 1 | 1 |
| | b2 | 1 | 1 | 1 | 0 | 0 | 0 |
| | c1 | 0 | 1 | 0 | 1 | 1 | 0 |
| | c2 | 1 | 0 | 1 | 0 | 0 | 1 |

**Table 3-6 - Example 2 - Data objects' complements matrix**

| Attribute Values | | Data Objects | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| | A1 | **2.4142** | **2.4142** | **2.4142** | **2.4142** | **2.4142** | **2.4142** |
| | A2 | 0 | 0 | 0 | 0 | 0 | 0 |
| | B1 | 2.3738 | 2.0404 | 2.3738 | 1.0404 | 1.0404 | 1.3738 |
| | B2 | 1.0404 | 1.3738 | 1.0404 | 2.3738 | 2.3738 | 2.0404 |
| | C1 | 2.3738 | 1.3738 | 2.3738 | 1.0404 | 1.0404 | 2.0404 |
| | C2 | 1.0404 | 2.0404 | 1.0404 | 2.3738 | 2.3738 | 1.3738 |

**Table 3-7 - Example 2 - Inner product between attributes similarity matrix and data objects**

| | | Data Objects | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| Attribute Values | A1 | 1.4142 | **1.4142** | 1.4142 | 1.4142 | 1.4142 | **1.4142** |
| | A2 | 0 | 0 | 0 | 0 | 0 | 0 |
| | B1 | 0.3333 | 0.6667 | 0.3333 | **1.6667** | **1.6667** | 1.3333 |
| | B2 | **1.6667** | 1.3333 | **1.6667** | 0.3333 | 0.3333 | 0.6667 |
| | C1 | 0.3333 | 1.3333 | 0.3333 | 1.6667 | 1.6667 | 0.6667 |
| | C2 | 1.6667 | 0.6667 | 1.6667 | 0.3333 | 0.3333 | 1.3333 |

**Table 3-8 - Example 2 - Inner product between attributes similarity matrix and data objects' complements**

| | | Data Objects | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| Attribute Values | a1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | a2 | 0 | 0 | 0 | 0 | 0 | 0 |
| | B1 | **2.0404** | **1.3738** | **2.0404** | -0.6262 | -0.6262 | 0.0404 |
| | B2 | -0.6262 | 0.0404 | -0.6262 | **2.0404** | **2.0404** | **1.3738** |
| | c1 | 2.0404 | 0.0404 | 2.0404 | -0.6262 | -0.6262 | 1.3738 |
| | c2 | -0.6262 | 1.3738 | -0.6262 | 2.0404 | 2.0404 | 0.0404 |

**Table 3-9 - Example 2 - Difference between the inner product of the attribute similarity matrix & data objects and the inner product of the attribute similarity matrix & data objects' complements**

## 3.6 Discussion of the objective function

In this section, we discuss the objective function used in the cluster assignments. The goal is to assign a data object to a cluster representative that is most similar to the data object and least similar to its complement. Measuring the similarity of a complement to a cluster representative is the same as measuring the dissimilarity of the data object to the

cluster representative. Our objective function assigns negative weights to all attribute categories that are not present in the data object which in turn measures the dissimilarity between a data object and a cluster representative. The similarity part of the function assigns positive weights to the attribute categories present in the data object. The definition of our function is inspired by the Condorcet's criterion (Marcotorchino and Michaud, 1979) which combines two factors in the criterion to provide a ranking. Specifically, it measures the intra-class similarity and inter-class dissimilarity and seeks to maximize both as follows (Maimon and Rokach, 2005):

$$\sum_{\substack{X^i, X^j \in C_i \\ X^i \neq X^j}} S(X^i, X^j) + \sum_{\substack{X^i \in C_i \\ X^j \notin C_i}} D(X^i, X^j)$$

Where $X^i$ and $X^j$ are data objects and $C_i$ is the $i^{th}$ cluster. The functions $S(X^i,X^j)$ and $D(X^i,X^j)$ measure the similarity and dissimilarity, respectively, between the data objects. In our case, we are ranking each row in the attribute similarity matrix with respect to a data object and its complement. Each data object and its complement can be considered as separate clusters. To rank each row in the matrix, we measure the similarity of the row to the data object and the dissimilarity of the row to the data object's complement. The dissimilarity of the complement is the negative of its similarity to any given row of the attributes similarity matrix.

The K-representatives algorithm (San et al., 2004) uses a dissimilarity function which indirectly utilizes a data object complement. In the K-representatives algorithm, the dissimilarity function is defined as follows:

$$D(X^i, Q_j) = \sum_{l=1}^{d} (1 - f_{al})$$

where $f_{al}$ is the relative frequency of attribute value $a_l$ in the $j^{th}$ cluster and $a_l$ is an attribute category that occurs in $X^i$. $Q_j$ is a vector of length m and contains the relative frequency of each attribute category in the $j^{th}$ cluster. The above function can be re-written using a data object complement as follows:

$$D(X^i, Q_j) = Q_j E^i$$

where $E^i$ is an m-dimensional vector that is the complement of the data object $Y^i$. $Y^i$ is the binary representation of the data object $X^i$. Note that our definition of a cluster representative is different from the cluster representative defined in San et al. (2004). We do not use relative frequency as defined in San. et al. (2004).

It is possible to get duplicate values of the function for two separate cluster representatives. In this case, we assign the data object to the cluster representative with the highest similarity to the data object disregarding the similarity of the cluster candidate to the data object complement.

To represent our function mathematically, we use $S(B^i,Q)$. $S(B^i,Q)$ which measures the similarity between the $i^{th}$ data object and each row of matrix Q. Here, Q is a matrix of cluster representatives. Matrix B contains the difference between the data objects and their complements matrices. If $B^i$ denotes a column in B and $Q_j$ is a cluster representative in Q, S is defined as

$$S(B^i, Q_j) = Q_j B^i$$

Alternatively, if $F_j$ is a vector of the occurrence frequency of all attribute categories in the $j^{th}$ cluster, we can write S as

$$S(B^i, Q_j) = \frac{F_j B^i}{\| F_j \|}$$

For a cluster, the function can be computed as follows:

$$S(B, Q_j) = \sum_{i=1}^{n_j} Q_j B^i$$

Where matrix B contains the data objects in the $j^{th}$ cluster and $n_j$ is the number of data objects in B. Alternatively, when the cluster representative is defined as the normalized vector of attribute categories occurrence frequencies (see Section 3.3), the function can be written as:

$$S(B, Q_j) = \left(2F_j - n_j e\right) Q_j^t$$

$$S(B, Q_j) = \frac{\left(2F_j - n_j e\right) F_j^t}{\| F_j \|}$$

$$S(B, Q_j) = \frac{\left(2F_j^2 - n_j F_j\right)}{\| F_j \|}$$

For the entire data set the function can be written as

$$\sum_{j=1}^{k} \frac{\left(2F_j^2 - n_j F_j\right)}{\| F_j \|}$$

Maximizing this function is similar to minimizing the sum of the squares of error (SSE). SSE for the $j^{th}$ cluster can be calculated as follows:

$$SSE = \sum_{i=1}^{n_j} \left( Y_i^t - \frac{F_j}{n_j} \right)^2$$

$$SSE = dn_j - \frac{F_j^2}{n_j}$$

Where $Y_i$ represents a data object. SSE for the data set is

$$dn - \sum_{j=1}^{k} \frac{F_j^2}{n_j}$$

## 3.7 An Iterative Process

In the previous sections, we discussed how each data object is assigned a candidate cluster from an initial set of candidate clusters' representatives. Based on empirical results for several data sets, the first assignment usually yields high quality clusters. However, we incorporate into the algorithm an iterative process that repeats the assignment of the data objects to newly recalculated cluster representatives using the objective function defined in section 3.6 until a fixed point is reached (no change of cluster assignments).

## 3.8 Cluster Merging

Cluster merging is another step included in the algorithm whose sole purpose is to reduce the number of clusters if desired. The merge algorithm merges clusters based on the cosine similarity of their cluster representatives. The input to the merge algorithm is the minimum number of desired clusters and a merge threshold. The purpose of the first input parameter is to determine if the merge algorithm should be executed. The merge

algorithm will not be executed when the current number of clusters is less than or equal to the minimum number of desired clusters. The merge threshold is used to determine which clusters can be merged. Clusters with a similarity above the threshold qualify for merging. The merge algorithm starts with the pair of clusters that has the highest similarity and iteratively merges with this pair all other clusters that are most similar to theses clusters (and other clusters that have been merged with these clusters). Once there are no more clusters to merge with the current set, the algorithm selects the next most similar pair of clusters that have not been processed and whose similarity is above the merge threshold. The process is repeated until there are no more remaining clusters. The threshold can be used to set the minimum similarity allowed between sub-clusters within a cluster. A threshold of one would merge only identical clusters. The merge algorithm is executed in every iteration within the clustering algorithm.

## 3.9 Step by Step Description of Algorithm

This section first gives an overview of the algorithm to highlight the main steps. A detailed description of the algorithm follows the overview. A simple example is also presented in this section to illustrate the algorithm.

### 3.9.1 Overview

1. Form the attribute categories similarity matrix using the cosine similarity measure (matrix T).
2. Define a matrix B that contains the difference between the data objects and the complements matrices.

3. Compare each data object and its complement to each row in the attributes categories similarity matrix (inner product of matrices T (or matrix Q in subsequent iterations) and B). Assign each data object to a row (cluster) of matrix T that has the highest inner product. Check if the cluster assignments have changed. If there is no change then the algorithm stops.
4. Merge the clusters.
5. Calculate new cluster representatives (matrix Q). Steps 3, 4, and 5 are repeated until cluster assignments do not change.

A pseudo-code of CATS appears in **Figure 3-1.**

```
%===== Normalize each row of matrix Y
D=Y
For each Y_i
      Y_i = Y_i /(Norm(Y_i))

%===== Compute attribute categories similarity matrix
T=Y*Y^t
Q=T

%===== D is the data objects matrix, Dcom is its complement
 B=D-Dcom

While (cluster assignments are changing)
            Q=Q*B
            Assign clusters
            Merge clusters
            Q = Normalized cluster representatives calculated using
                the occurrence frequency of  each attribute category
                in each cluster
 End
```

**Figure 3-1 – Pseudo-code for CATS**

### 3.9.2 Detailed Description

The following implementation is for illustration purposes only. For an efficient implementation, the attribute categories cosine similarity matrix can be calculated without materializing the data objects matrix into a binary matrix.

**Step 1: Collect the summary data**

The attributes categories similarity matrix is computed using normalized binary vectors $Y_i$. Any rows or columns that contain only zeros (i.e., attribute categories that are possible but do not occur in the given data set) are removed. We initialize the matrix Q to the attribute categories similarity matrix.

**Step 2: Define a matrix that contains the data objects and their complements**

Instead of creating two matrices, one for the data objects and the other for the complements, we create one matrix to contain both the data objects and their complements. In a copy of the data objects matrix, we replace zeros with –1 to give negative weights to missing values and to represent the complements. This matrix will be used to compute the dot product with each row of the cluster representatives' matrix (matrix Q). Note that this matrix is equal to the difference between the data objects matrix and its complement. We will call this matrix B.

**Step 3: Assign each data object to a cluster by comparing each data object and its complement to each row in the cluster representatives' matrix**

In this step, we calculate the dot product of each data objects and its complement (matrix B) and each row of the cluster representatives' matrix (matrix Q). We assign each data object to the row in matrix Q with the highest weight (cluster assignments). If there is no change in cluster assignments, the algorithm stops.

**Step 4: Merge the clusters**

In this step, clusters are merged based on the cosine similarity of their representatives. Please see Section 3.8 for a detail description.

**Step 5: Calculate new cluster representatives**

In this step, we calculate the new cluster representatives as a vector of the frequency of each attribute category in the cluster. We normalize each cluster representative. Matrix Q will contain the new cluster representatives. Steps 3, 4, and 5 are repeated until cluster assignments are not changing.

### 3.9.3 Example 3

To illustrate how cluster representatives are created, we will use the data presented in Example 2. Based on the data presented in Table 3-9, the data objects in Example 2 are clustered into two groups based on their weights with respect to the initial cluster representatives (rows in the attribute categories similarity matrix). Data objects 1, 2 and 3 are placed in one cluster while data objects 4, 5, and 6 form the second cluster. Two cluster representatives will be created. Each cluster representative is an m-vector that contains the occurrence frequency of each attribute category in a cluster. Table 3-10 shows the occurrence frequency in each cluster. Table 3-11 shows the cluster representatives. Table 3-12 shows the inner product between the cluster representatives and the data objects and their complements. Based on the data in Table 3-12, the cluster assignments do not change, and CATS terminates.

| | | Attribute Categories - Occurrence Frequency | | | | | |
|---|---|---|---|---|---|---|---|
| Clusters | | 1 | 2 | 3 | 4 | 5 | 6 |
| | 1 | 3 | 0 | 3 | 0 | 2 | 1 |
| | 2 | 3 | 0 | 0 | 3 | 1 | 2 |

**Table 3-10 – Example 3 - Attribute categories occurrence frequency**

| | | Attribute Categories | | | | | |
|---|---|---|---|---|---|---|---|
| Clusters' Representatives | | 1 | 2 | 3 | 4 | 5 | 6 |
| | 1 | 0.6255 | 0 | 0.6255 | 0 | 0.417 | 0.2085 |
| | 2 | 0.6255 | 0 | 0 | 0.6255 | 0.2085 | 0.417 |

**Table 3-11 – Example 3 - Clusters' representatives**

| | | Data Objects | | | | | |
|---|---|---|---|---|---|---|---|
| Clusters | | 1 | 2 | 3 | 4 | 5 | 6 |
| | 1 | 1.4596 | 1.0426 | 1.4596 | -0.2085 | -0.2085 | 0.2085 |
| | 2 | -0.2085 | 0.2085 | -0.2085 | 1.4596 | 1.4596 | 1.0426 |

**Table 3-12 – Example 3- Difference between the inner product of the clusters' representatives & data objects and the inner product of the clusters' representatives & data objects' complements**

## 4 Analysis of CATS

In this section, we present the analysis of CATS in terms of I/O and CPU costs. As defined in Section 3.1, n is the number of data objects, m is the number of attribute categories, d is the number of attributes. Let k be the average number of clusters over all iterations and j be the number of iterations.

- **Compute the attribute values similarity matrix.**

  This step requires one pass through the data to calculate the co-occurrence frequency of each pair of attribute values and the frequency of each attribute category. The I/O cost is the cost of reading the data from the disk. The CPU cost entails incrementing the counters and is $O(n\frac{m^2}{2})$.

- **Cluster Assignments (loop)**

In this step, the weight of each data object and a cluster representative is calculated and as such the CPU cost will be O(njk). Note that the data objects matrix can be stored as a sparse matrix and therefore we do not need to materialize all entries for the data objects.

- **Merge Algorithm**

  The merge algorithm merges the cluster representative using cosine similarity. The CPU cost will depend on the average number of clusters and number of iterations $O(jk^2)$. Based on experimental results, the number of clusters shrinks by more than 50% after the first iteration of the loop and more than 75% after the second iteration (see Table 3-15).

The theoretical time complexity of CATS is linear in terms of the number of objects and is comparable to the K-representatives algorithms. CATS incurs additional time for the construction of the attribute similarity matrix and the merge algorithm but, based on empirical results, the bulk part of the execution time is incurred during the cluster assignment phase.

## 5 Experimental Evaluation

We have developed a MATLAB® implementation to evaluate the cluster quality and scalability of CATS. Tests with our implementation on several UCI Machine Learning Repository (Asuncion and Newman 2007) data sets that are used extensively to evaluate the quality of categorical clustering algorithms show that CATS produces highly accurate clusterings. We compare the cluster quality of CATS with several well-known clustering algorithms for categorical data. We also show how the algorithm scales to large data sets using synthetic data.

## 5.1 Algorithms Used in the Comparative Analysis

**CATS**

This is the algorithm presented in this chapter. It is an iterative process that uses a deterministic method to initialize the cluster representatives.

**K-representatives**

The K-representative algorithm is an extended version of the K-means algorithm which is customized for categorical data. For a detailed description, please see Chapter 2.

**CLICKS**

Zaki et al. (2007) presented CLICKS which defines the clustering problem in terms of the maximal cliques in a K-partite graph. For a more detailed description, please see Chapter 2.

**COOLCAT**

Based on a sample of data object selected randomly from the data set, COOLCAT (Barbara et al, 2002) selects k dissimilar data objects. These selected data objects become cluster seeds for k clusters. COOLCAT assigns the rest of the data objects to any of the k clusters based on cluster entropy. For a more detailed description, please see Chapter 2.

**LIMBO**

Similar to COOLCAT, LIMBO (Andritsos et al., 2004) uses information entropy to create the clusters. LIMBO is the first scalable agglomerative hierarchical entropy-based clustering algorithm for categorical data. For a more detailed description, please see Chapter 2.

**Rock**

ROCK (Guha et al, 1999) is a hierarchical clustering algorithm based on the concept of links between data objects. For a more detailed description, please see Chapter 2.

## 5.2 Comparative Measures

Two measures are used to compare CATS with other algorithms. The first is cluster accuracy and is calculated as follows (Huang, 1998):

$$\text{Accuracy} = \frac{1}{n}\sum_{l=1}^{k} h_l$$

where $h_l$ is the number of data objects properly classified (appear in cluster *l* and in its corresponding label—each data object is assigned a label that reflects the pre-defined class that the data objects belongs to), n is the number of data objects in the data set and k is number of clusters. Note that if more clusters are constructed than the number of labels, then a label can be assigned to more than one cluster for the purpose of calculating the accuracy of the clustering but a cluster can only be assigned to one label.

Entropy is another measure used in calculating the quality of the clusters. Entropy of a cluster, denoted by E, is defined as the disorder within the cluster. Information entropy measures the uncertainty of an outcome (Shannon, 1948). Cheng et al. (2006) used it in measuring clustering quality. Lower entropy implies better clustering quality. It is defined as follows:

$$E(C_j) = \sum_{i=1}^{k} -\left(\frac{|L_i \bigcap C_j|}{|C_j|}\right)\log\left(\frac{|L_i \bigcap C_j|}{|C_j|}\right)$$

where $L_i$ is the known class label and $C_j$ is the cluster generated by the algorithm. |..| refers to the cardinality of the set. The lower the entropy, the better the clustering will be.

## 5.3 Standard Data sets

For our evaluation, we use standard test data sets that have been widely used in literature for clustering categorical data. The data sets, which are provided in the UCI machine learning repository (Asuncion and Newman, 2007), are the Soybean, Congressional Votes, and Mushroom data sets. Test results on these data sets are available for most of the categorical clustering methods discussed in this thesis.

**Soybean data set**

The Soybean data set contains 47 data objects with 35 categorical attributes that describe 4 categories of soybean diseases. Only 21 of 35 attributes have more than one value and as such only these 21 attributes are used. Three of the clusters contain 10 data objects each; the fourth cluster contains 17 data objects.

**Congressional Votes data set**

The Congressional Votes data set contains 435 data objects with 16 categorical attributes. The data set contains the 1984 United States congressional vote's records. Each data object is classified as either democrat (267) or Republican (168). Each attribute can have a value of "yes" or "no". The data set has some missing data (undecided votes). One data object did not contain any values (all values were missing) and as such the data object was excluded from the analysis.

**Mushroom data set**

The Mushroom data set contains 8,124 data objects classified as edible or poisonous mushrooms. The data set has 22 categorical attributes with multiple values in the domain. There are 4,208 data objects in the edible class and 3,916 in the poisonous class. In the data set, 2,480 attribute values are missing.

## 5.4 Synthetic Data

To produce synthetic data sets, we use a data generator available on the web[4] and used by other researchers for testing clustering algorithms (Andritsos et al., 2004). This data generator has several parameters that allow a user to specify the characteristic of an output data set such as the number of attributes, attributes' domain size, number of clusters, noise level, number of attributes included in a rule, and the size of the data set. The rules used in this data generator are in conjunctive normal form (*attribute1=valueA* ***and*** *attribute2=valueB* ➔ class 1).

## 5.5 Results and Discussion

### 5.5.1 Standard Data Sets

In this section, we first present the results of CATS using the standard data sets to examine the clustering accuracy of the algorithm and the effect of the merge threshold on the clustering granularity. We, then present comparative results for CATS on the same data sets. Last, we present the evaluation of the algorithm in terms of cluster quality and scalability on synthetic data.

Table 3-13 shows the clustering accuracy with merge thresholds values of 50% and 90% for test runs on the standard data sets: Soybean, Congressional Votes, and Mushroom. The clustering accuracy values are 88% and 89% with 2 clusters for the Congressional Votes and Mushroom data sets with a merge threshold of 0.5. Higher clustering accuracy is achieved with higher merge threshold but the number of clusters also increases. The clustering accuracy is 100% with 23 clusters for the Mushroom data

[4] www.datgen.com.

set; and 91% with 4 clusters (merge threshold is 0.9) or 94% with 8 clusters (merge threshold is 0.95) for the Congressional Votes data set. As for the Soybean data set, the clustering accuracy is 100% in all cases for 4 clusters regardless of the merge threshold. As Table 3-13 shows, generally, the higher the merge threshold, the more clusters the algorithm finds and the better the clustering quality.

| Data set Name | Number of iterations | Number of clusters | Accuracy | Merge threshold | Remarks |
|---|---|---|---|---|---|
| **Soybean** | 3 | 4 | **100%** | 0.9 | |
| | 3 | 4 | **100%** | 0.5 | |
| **Congressional Votes** | 6 | 4 | **91%** | 0.9 | CATS achieves an accuracy of **94%** for 8 clusters if a merge threshold of .95 is used |
| | 4 | 2 | 88% | 0.5 | |
| **Mushroom** | 5 | 23 | **100%** | 0.9 | |
| | 19 | 2 | 89% | 0.5 | |

**Table 3-13 - Accuracy results for Soybean, Congressional Votes and Mushroom data sets**

We also compared CATS to the K-representatives algorithm. Table 3-14 shows the results for 10 runs of the K-representatives algorithm on the Soybean data set with an average cluster accuracy of 85%. CATS achieves a clustering accuracy of 100% for the Soybean data set.

| | Number of Iterations | Cluster Accuracy |
|---|---|---|
| | 9 | 100.00% |
| | 4 | 100.00% |
| | 5 | 78.72% |
| | 7 | 78.72% |
| | 4 | 100.00% |
| | 3 | 57.45% |
| | 5 | 78.72% |
| | 5 | 100.00% |
| | 5 | 100.00% |
| | 3 | 57.45% |
| **Average** | 5 | 85.11% |

**Table 3-14 - K-representative algorithm on the Soybean data set**

Next, we take a closer look at how the number of intermediate clusters changes and how fast it converges to the final number of clusters. Table 3-15 shows the results for all standard data sets tested in this chapter. By the second iteration for all data sets, the number of clusters is decreased by close to 75% and high levels of clustering accuracy are achieved by the first iteration. Figure 3-2, Figure 3-3, and Figure 3-4 show the relation between the number of intermediate clusters & clustering accuracy versus the number of iterations for the Soybean, Congressional Votes, and Mushroom data sets, respectively. Note that iteration number 0 is included just to show the starting number of clusters seeds.

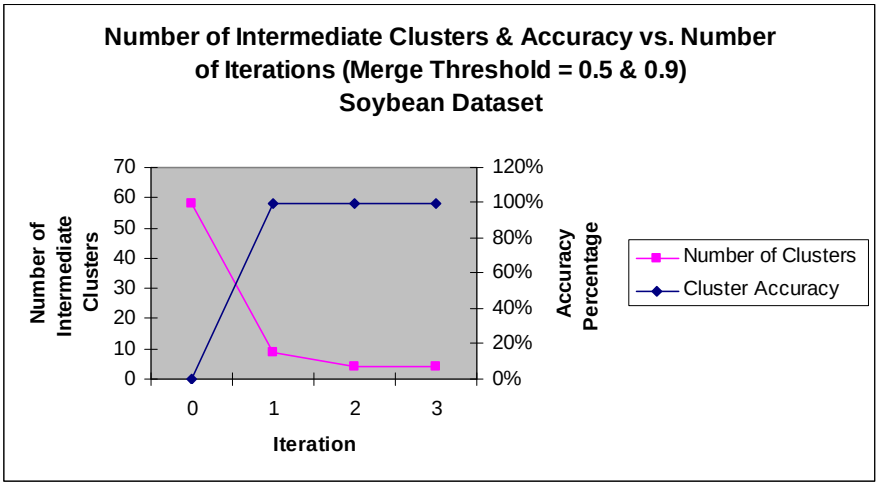


**Figure 3-2 - Soybean data set - Number of intermediate clusters**

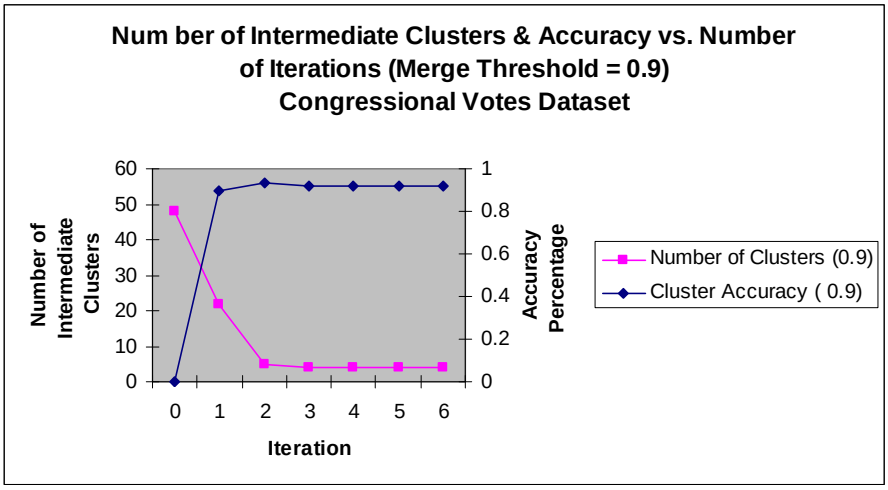


**Figure 3-3 - Congressional Votes data set - Number of intermediate clusters**

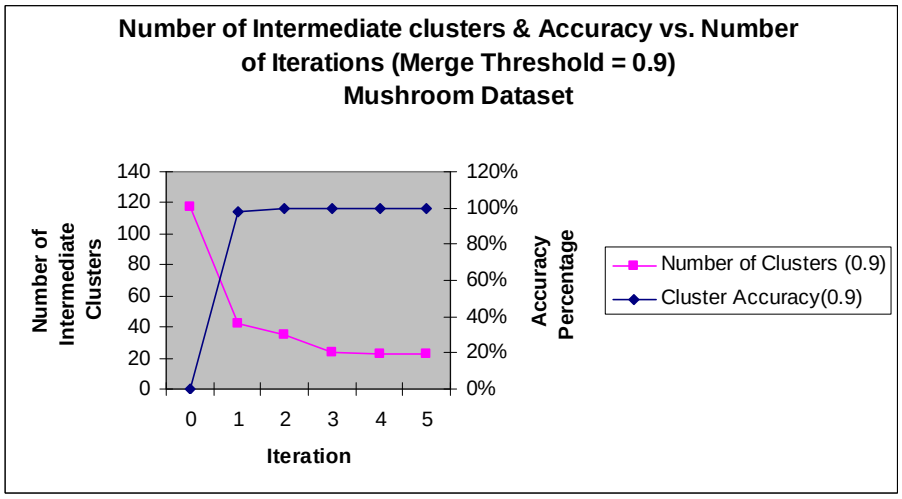


**Figure 3-4 - Mushroom data set - Number of intermediate clusters**

| Iteration | Soybean | | | | Congressional Votes | | | | Mushroom | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Number of clusters (0.5) | % of change in Number of Clusters | Cluster Accuracy (0.5) | Number of clusters (0.9) | % of change in Number of Clusters | Cluster Accuracy (0.9) | Number of clusters (0.5) | % of change in Number of Clusters | Cluster Accuracy (0.5) | Number of clusters (0.9) | % of change in Number of Clusters | Cluster Accuracy (0.9) |
| 0 | 58 | 0% | 0% | 58 | 0% | 0 | 48 | 0% | 0% | 48 | 0% | 0% | 117 | 0% | 0 | 117 | 0% | 0% |
| 1 | 9 | 84% | 100% | 9 | 84% | 100% | 22 | 54% | 88% | 22 | 54% | 90% | 42 | 64% | 97% | 42 | 64% | 98% |
| 2 | 4 | 93% | 100% | 4 | 93% | 100% | 2 | 96% | 88% | 5 | 90% | 93% | 18 | 85% | 90% | 35 | 70% | 100% |
| 3 | 4 | 93% | 100% | 4 | 93% | 100% | 2 | 96% | 88% | 4 | 92% | 92% | 4 | 97% | 52% | 24 | 79% | 100% |
| 4 | | | | | | | 2 | 96% | 88% | 4 | 92% | 92% | 2 | 98% | 52% | 23 | 80% | 100% |
| 5 | | | | | | | | | | 4 | 92% | 92% | 2 | 98% | 55% | 23 | 80% | 100% |
| 6 | | | | | | | | | | 4 | 92% | 92% | 2 | 98% | 60% | | | |
| 7 | | | | | | | | | | | | | 2 | 98% | 62% | | | |
| 8 | | | | | | | | | | | | | 2 | 98% | 62% | | | |
| 9 | | | | | | | | | | | | | 2 | 98% | 67% | | | |
| 10 | | | | | | | | | | | | | 2 | 98% | 69% | | | |
| 11 | | | | | | | | | | | | | 2 | 98% | 69% | | | |
| 12 | | | | | | | | | | | | | 2 | 98% | 69% | | | |
| 13 | | | | | | | | | | | | | 2 | 98% | 70% | | | |
| 14 | | | | | | | | | | | | | 2 | 98% | 72% | | | |
| 15 | | | | | | | | | | | | | 2 | 98% | 76% | | | |
| 16 | | | | | | | | | | | | | 2 | 98% | 85% | | | |
| 17 | | | | | | | | | | | | | 2 | 98% | 89% | | | |
| 18 | | | | | | | | | | | | | 2 | 98% | 89% | | | |
| 19 | | | | | | | | | | | | | 2 | 98% | 89% | | | |

**Table 3-15 - Number of intermediate clusters for the Soybean, Congressional Votes, and Mushroom data sets**

Table 3-16 and Table 3-17 present clustering accuracy and entropy of the algorithm and other comparable algorithms listed in Section 5.1 for the Congressional Votes and Mushroom data sets. Note that the results shown in these tables are for the case where the number of clusters is restricted to the published number of clusters for these data sets. (Please see the remarks column in Table 3-16 and Table 3-17 for accuracy levels when the number of clusters is not restricted to the number pre-defined classes.) For the Congressional Votes data set, CATS achieves the highest accuracy level and lowest entropy (88% and .45, respectively) of all algorithms presented. With the exception to LIMBO, CATS produces more accurate clusters than all other algorithms presented in the table for the Mushroom data set (accuracy level of 89%).

| | Accuracy | Entropy | Remarks |
|---|---|---|---|
| **CATS** | 88% | 0.452 | This is the result for 2 clusters. For 8 clusters, the accuracy level is 94% |
| **K-representatives** | 88% | | This is average of 10 executions of the algorithm using random partitioning. |
| **Traditional Hierarchical Clustering Algorithm** | 86% | | 157 and 215 records were properly classified (Guha et al., 1999) |
| **Rock** | 79% | 0.499 | 144 and 201 records were properly classified. 62 records were excluded as they were considered outliers and were not classified (Guha et al., 1999). We use a data set size of 434 when we calculate the accuracy level for ROCK to provide comparability with other algorithms presented.<br>Entropy results were obtained from Cheng et al., 2006. |
| **LIMBO** | 87% | | Results obtained from Andritsos et al., 2004. |
| **COOLCAT** | 85% | 0.498 | Accuracy results are from Andritsos et al., 2004.<br>Entropy results are from Cheng et al., 2006. |
| **CLICKS** | Not Available for 2 clusters | Not Available for 2 clusters | An accuracy level of 96% is achieved with 13 clusters. One of the 13 groups is designated for outliers. |

**Table 3-16- Results for the Congressional Votes Data set (2 clusters)**

| | Accuracy | Remarks |
|---|---|---|
| **CATS** | 89% | This is the accuracy level for 2 clusters. An accuracy level of **100%** is achieved with 23 clusters. |
| **K-representative** | 78% | This is average of 10 executions of the algorithm using random partitioning. |
| **Traditional Hierarchical Clustering Algorithm** | 53% | These are the results for 20 clusters(Guha et al., 1999) |
| **ROCK** | 57% | This is the clustering result with 2 clusters (Andritsos et al., 2004). Guha et al. (1999) implement Rock to partition the data set into 21 clusters but could not further combine the clusters to form the two pre-defined classes; only 32 records were not clustered properly. |
| **LIMBO** | 89% | Accuracy results for 2 clusters obtained from Andritsos et al, 2004. |
| **COOLCAT** | 73% | Accuracy results for 2 clusters are from Andritsos et al, 2004. |
| **CLICKS** | Not Available for 2 clusters | An accuracy level of 97% is achieved with 19 clusters (Zaki et al., 2007). |

**Table 3-17 -Results for the Mushroom Data set (2 clusters)**

### 5.5.2 Synthetic Data Sets

In this section, we present the results of several experimental evaluations of the algorithm using synthetic data. We will show comparative results for cluster quality as well scalability using synthetic data sets.

**Cluster Quality**

We have implemented the K-representative algorithm in MATLAB® for the purpose of comparing its clustering quality with the results of CATS. We have constructed five data sets with various characteristics as follows:

| Data Set Name | Domain size of attributes | Number of Attributes | Number of attributes that participate in a rule | Number of Clusters | Data set Size | Noise Ratio |
|---|---|---|---|---|---|---|
| DS1 | 10-20 | 10 | 4 | 5 | 1000 | 0 |
| DS2 | 10-20 | 10 | 4 | 10 | 5000 | 0 |
| DS3 | 10-20 | 10 | 5 | 10 | 5000 | 2% |
| DS4 | 10-20 | 10 | 5 | 10 | 5000 | 10% |
| DS5 | 10-20 | 20 | 10 | 10 | 5000 | 10% |

**Table 3-18 Synthetic data sets**

For clustering accuracy of the K-representative algorithm, we executed the algorithm ten times and calculated the average cluster accuracy for each data set. We initialized the clusters using random partitioning method. To make the two algorithms comparable, we set the input parameter of the merge step (minimum number of clusters) of CATS to the number of expected clusters in the data set. Table 3-19 shows comparative results of CATS and the K-representative algorithm for the five synthetic data sets:

| | CATS | | | | K-Representatives | |
|---|---|---|---|---|---|---|
| Data Set Name | Merge Threshold | Cluster Accuracy | Number of Clusters | Number of Iterations | Average cluster Accuracy | Average Number of terations |
| DS1 | 0.5 | 100% | 5 | 5 | | |
| | 0.9 | 99.90% | 46 | 12 | 96.45% | 5.12 |
| DS2 | 0.5 | 100% | 10 | 6 | | |
| | 0.9 | 100% | 10 | 8 | 91.11% | 11.7 |
| DS3 | 0.5 | 99.96% | 10 | 7 | | |
| | 0.9 | 99.94% | 11 | 10 | 91.16% | 9.9 |
| DS4 | 0.5 | 99.90% | 10 | 7 | | |
| | 0.9 | 98.60% | 12 | 19 | 95.87% | 8.6 |
| DS5 | 0.5 | 90.42% | 10 | 7 | | |
| | 0.9 | 100% | 10 | 7 | 90.23% | 8.4 |

**Table 3-19 Comparative results with K-representatives algorithm**

In terms of cluster quality, Table 3-19 shows that our algorithm outperformed the K-representative algorithm on all data sets. The theoretical time complexity of CATS is linear in terms of the number of data objects and is comparable to the K-representatives algorithms. The actual time complexity for each algorithm can be compared in terms of the number of iterations performed by each algorithm. With a merge threshold of 0.5, CATS clustered the data using less number of iterations than the K-representative algorithm for all data sets. From a practical point, CATS outperformed the K-representative in terms of execution time since the K-representative algorithm needs to be executed multiple times whereas CATS need not.

**Size of the Data Set**

To find out how the execution time of CATS varies as the data set size varies, we created 10 synthetic data sets of increments of 6,500 with 10 attributes each and a domain size of 10 to 20 values for each attribute. Each of these data sets contains five clusters. Figure 3-5 shows how the elapsed time changes as the data set size changes. Figure 3-6 plots the data set size and the ratio of change in the elapsed time. Both figures illustrate that the execution time is linear in terms of the data set size. Note that the actual elapse time per se should not be taken as a benchmark for the execution time of CATS as execution time varies based on the type of CPU, memory, implementation, and of course programming language. These figures, though actual for our implementation, are just for illustrating the relationship between the data set size and the elapsed time.

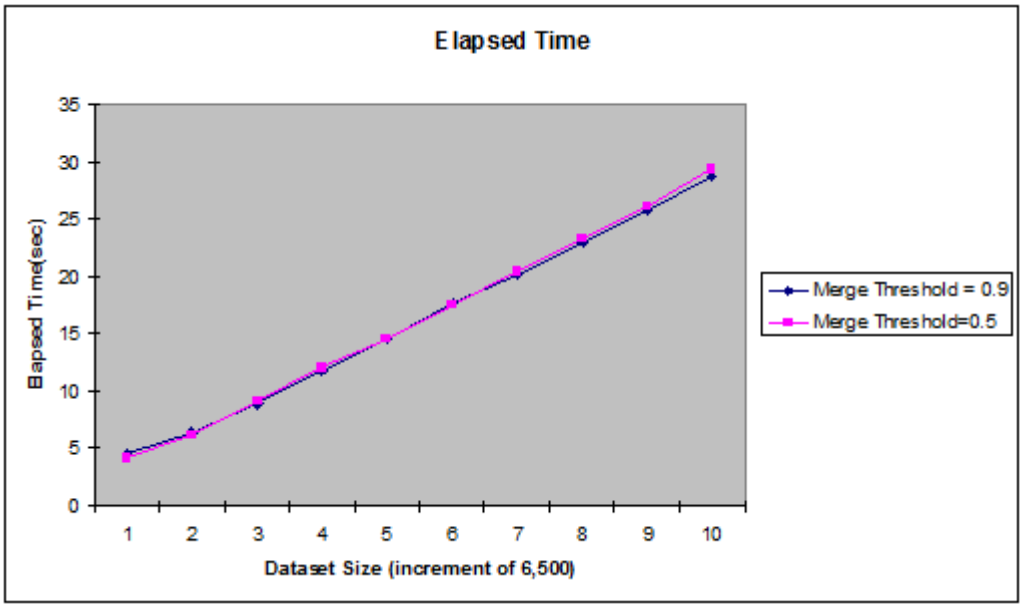


**Figure 3-5 – Data set Size vs. Elapsed Time for CATS**

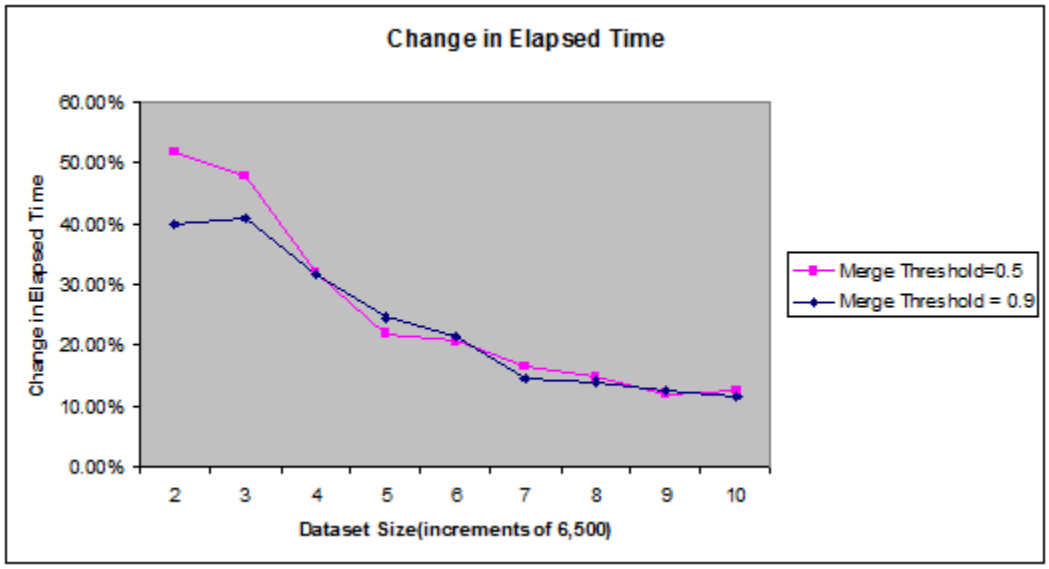


**Figure 3-6 - Data set Size vs. Change in Elapse Time for CATS**

### Number of Attributes

Next, we wanted to investigate how the execution time varies as the number of attributes increases. We created 5 synthetic data sets with 5,000 data objects and 5 clusters in each data set. The first data set has 5 attributes; the second data set has 10 attributes; the third data set has 15 attributes; the fourth data set has 20 attributes; and the last data set has 30

attributes. The domain size of each attribute is 10. Figure 3-7 shows a linear relationship between the number of attributes and the elapsed time while the clustering accuracy remained at 100%.

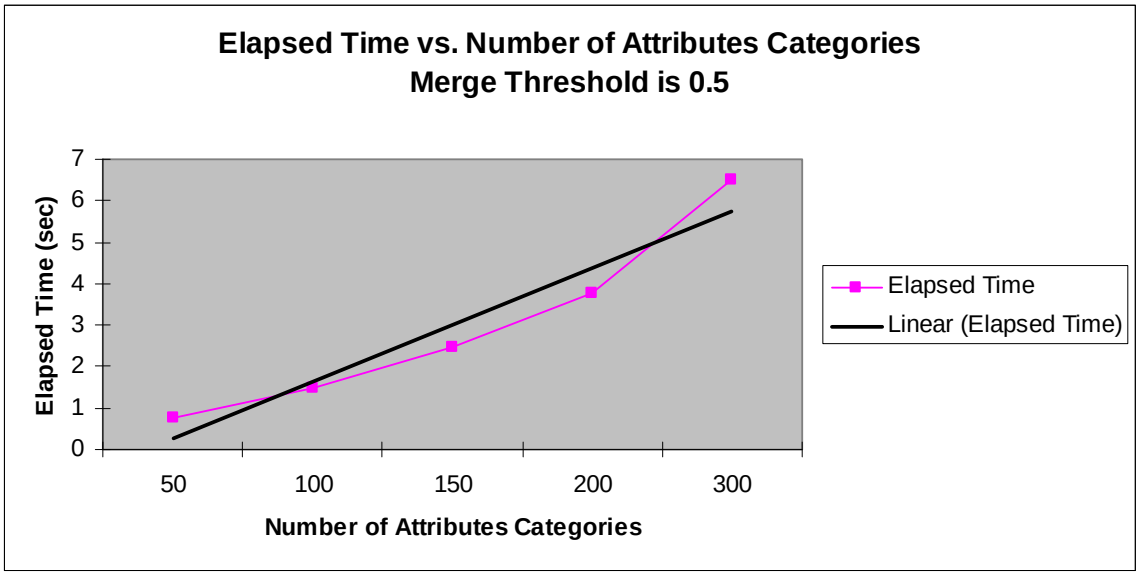


**Figure 3-7 - Elapsed Execution Time Vs. Number of Attributes Categories**

## 6 *Data set Suitability*

CATS clusters the data objects based on their weights with respect to each of the candidate clusters. However, when the candidate clusters are not sufficiently distinguishable from each other, (i.e each attribute is uniformly distributed across all other attributes), CATS may not produce the desired clusters. For example, we ran CATS on the Lens data set which is obtained from UCI Machine learning repository (Asuncion and Newman, 2007). CATS achieved a clustering accuracy of only 63% (the K-representatives algorithm achieves similar results). The Lens data set has 24 rows and 3 clusters. All attribute categories in the Lens data set co-occur uniformly with all other attribute categories as shown in Table 3-20. Table 3-20 shows the co-occurrence

frequency of each pair of values. The Lens data set has 4 attributes identified on Table 3-20 as A, B, C, and D. The domain size of attribute A is 3 and the domain size of the rest of the attributes is 2. The mean and the standard of deviation of the distribution of co-occurrence frequency is the same for all attributes with the same domain size. Table 3-21 shows the attribute categories similarity matrix, mean and standard of deviation of the distribution of all attribute categories. Note that the mean and standard of deviation are the same for all categories in attributes B, C, D (domain size is 2). As for attribute A, the distribution parameters are the same for all values in the domain. In the case of the Lens data set, the data objects clustered based on the categories in attribute A. CATS produced a clustering with a lower sum of the squares of error (SSE) than the known clustering for the data set. For the Lens data set, SSE of CATS clustering is 36 whereas the known clustering of the data set has an SSE of 39.20. In all of the experiments presented in this thesis, we do not add the classification attribute (an attribute that contains a label identifying the cluster), but for the Lens data set, we ran an additional test to include the classification attribute since it was the only attribute not uniformly distributed across all other attributes. The result of this test on the Lens data set is a clustering accuracy of 100% with 3 clusters.

| | | Attributes Values | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | A1 | A2 | A3 | B1 | B2 | C1 | C2 | D1 | D2 |
| Attributes Values | A1 | 8 | 0 | 0 | 4 | 4 | 4 | 4 | 4 | 4 |
| | A2 | 0 | 8 | 0 | 4 | 4 | 4 | 4 | 4 | 4 |
| | A3 | 0 | 0 | 8 | 4 | 4 | 4 | 4 | 4 | 4 |
| | B1 | 4 | 4 | 4 | 12 | 0 | 6 | 6 | 6 | 6 |
| | B2 | 4 | 4 | 4 | 0 | 12 | 6 | 6 | 6 | 6 |
| | C1 | 4 | 4 | 4 | 6 | 6 | 12 | 0 | 6 | 6 |
| | C2 | 4 | 4 | 4 | 6 | 6 | 0 | 12 | 6 | 6 |
| | D1 | 4 | 4 | 4 | 6 | 6 | 6 | 6 | 12 | 0 |
| | D2 | 4 | 4 | 4 | 6 | 6 | 6 | 6 | 0 | 12 |

**Table 3-20 – Lens Data set – Attribute Values Co-occurrence Frequency**

| | | Attributes Values | | | | | | | | | Mean | Standard of Dev |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A1 | A2 | A3 | B1 | B2 | C1 | C2 | D1 | D2 | | |
| Attributes Values | A1 | 1 | 0 | 0 | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.38 | 0.29 |
| | A2 | 0 | 1 | 0 | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.38 | 0.29 |
| | A3 | 0 | 0 | 1 | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.41 | 0.38 | 0.29 |
| | B1 | 0.41 | 0.41 | 0.41 | 1 | 0 | 0.5 | 0.5 | 0.5 | 0.5 | 0.47 | 0.25 |
| | B2 | 0.41 | 0.41 | 0.41 | 0 | 1 | 0.5 | 0.5 | 0.5 | 0.5 | 0.47 | 0.25 |
| | C1 | 0.41 | 0.41 | 0.41 | 0.5 | 0.5 | 1 | 0 | 0.5 | 0.5 | 0.47 | 0.25 |
| | C2 | 0.41 | 0.41 | 0.41 | 0.5 | 0.5 | 0 | 1 | 0.5 | 0.5 | 0.47 | 0.25 |
| | D1 | 0.41 | 0.41 | 0.41 | 0.5 | 0.5 | 0.5 | 0.5 | 1 | 0 | 0.47 | 0.25 |
| | D2 | 0.41 | 0.41 | 0.41 | 0.5 | 0.5 | 0.5 | 0.5 | 0 | 1 | 0.47 | 0.25 |

**Table 3-21 – Lens Data set – Attribute Values Similarity Matrix**

## 7 Remarks

In this chapter, we have proposed a novel deterministic method for initializing the clusters in a K-means like algorithm. CATS extends the concept of cluster representative presented in the K-representatives algorithm for categorical data. The results presented for CATS on standard and synthetic data sets are very competitive and encouraging.

CATS differs from K-means-like algorithms (such as K-modes and K-representatives) as follows:

1) CATS uses a deterministic approach to initialize the cluster seeds. The cluster seeds are initialized based on the attributes similarity matrix.
2) CATS is not sensitive to record order.
3) CATS automatically determines the number of clusters.
4) The merge algorithm provides a threshold to control the granularity of the clustering results (merge threshold).
5) CATS will produce the same results over multiple program executions and as such does not need to be executed multiple times to determine an optimal clustering.

# Chapter 4 – Spectral Based Algorithms Using Data Summaries

In this chapter, we combine the use of data summaries and spectral techniques to design clustering algorithms for categorical data that are scalable for high-dimensional data and large data sets. In Chapter 3, we introduced CATS, an iterative algorithm that uses data summaries to cluster categorical data. CATS does not use any dimension-reduction techniques to handle high-dimensional data. Thus, CATS may be computationally expensive for high-dimensional data (number of attributes). The algorithms presented in this chapter employ spectral relaxation techniques similar to those advanced in (Pothen et al., 1990; Roweis, 1997; Shi and Malik, 2000; Ng et al., 2001; Zha et al., 2001; Drineas et al., 2004; Kannan et al., 2004; Ding and He, 2004) to reduce the computational complexity inherent in clustering and also avoid the problem of convergence to local minima found in straightforward extensions of K-means to categorical clustering, e.g., K-modes.

## *1 Introduction*

In this chapter, we present two novel algorithms: SCCADDS1 and SCCADDS2 where SCCADDS stands for (***S***pectral-based ***c***lustering algorithm for ***ca***tegorical ***d***ata using ***d***ata ***s***ummaries). The algorithms have two main features. First, they use data summaries that consist of attribute occurrence and co-occurrence frequencies to create a set of vectors each of which represents a cluster. We refer to these vectors as "candidate cluster representatives." Second, the SCCADDS algorithms use spectral decomposition of the data summaries matrix to project and cluster the data objects in a reduced space. The

algorithms are designed for categorical data where relationship between data objects is induced by the relationship between attributes as Dhillon describes in (Dhillon, 2001) with respect to terms and documents.

In many aspects, categorical data sets are similar to term-document data sets that are the focus of information retrieval. Accordingly, our algorithms build on the theoretical framework of Latent Semantic Indexing (LSI) (Deerwester et al., 1990; Berry et al., 1995; Bradford, 2006), which has been successfully used in document indexing and retrieval. SCCADDS2 further improves on SCCADDS1 by incorporating the concepts of Principal Components Analysis (PCA) (Harris, 2001; Joliffee, 2002; Wall et al., 2003; Suhr, 2005).

SCCADDS differ from other clustering algorithms in several aspects (note: we use the term SCCADDS to refer to SCCADDS1 and SCCADDS2 where the discussion applies to both SCCADDS1 and SCCADDS2).

- Unlike existing spectral-based algorithms, SCCADDS use the attribute categories proximity matrix instead of the data object similarity matrix (as in most methods based on normalized cut of a graph of nodes representing data objects). This makes SCCADDS scalable for large data sets since in most categorical clustering applications the number of attribute categories is very small relative to the number of data objects.
- Non-recursive spectral-based clustering algorithms typically require K-means or some other iterative clustering method after the data objects have been projected into a reduced space. SCCADDS clusters the data objects directly by comparing

them to candidate cluster representatives without the need for an iterative clustering method.

- Unlike current spectral-based algorithms, SCCADDS is linear in terms of the number of data objects.
- SCCADDS requires only one parameter (the number of eigenvectors), which can be determined using measures such as inter and intra-cluster similarity. (This of course requires experimentation, but SCCADDS provides a capability for determining a suitable number of clusters.)
- The optional merge step of SCCADDS provides a mechanism to control the granularity of the clustering (small number of clusters vs. large number of clusters) without the need to specify an *exact* number of clusters as in most spectral-based clustering algorithms.
- SCCADDS is based on the theoretical framework used in Latent Semantic Indexing (LSI) and Principal Component Analysis (PCA).
- The clustering phase of SCCADDS can easily be programmed to take advantage of parallel computing resources to significantly reduce the algorithm's run-time (elapsed time) for large data sets.
- SCCADDS can be extended to handle fuzzy clustering, i.e., clusterings where an object may belong to more than one cluster based on membership weight.
- SCCADDS is easy to implement using off-the shelf linear algebra software.

Based on our experimental evaluations on standard and synthetic data sets, the accuracy of the clusters produced by the SCCADDS algorithms is competitive (and frequently

better than) the accuracy achieved by other spectral and non-spectral clustering algorithms.

Our discussion in section 2 begins with an overview of spectral data clustering. We next review the linear algebra needed to describe the SCCADDS algorithms and understand spectral clustering, particularly the Singular Value Decomposition and Eigen-decomposition. Then, we present the motivation behind spectral clustering and discuss how it provides a relaxed solution to the discrete clustering problem. We also discuss current spectral-based methods such as LSI and PCA. In Section 3, we describe the SCCADDS algorithms and present experimental evaluations of the algorithms. In Section 3, we also present experiments that illustrate the scalability of the algorithms in terms of data set size and number of dimensions (attributes). Finally, Section 4 offers some concluding remarks.

## 2 Background and Related Work for Spectral Clustering

Spectral clustering is a term used to characterize clustering methods that employ a few eigenvectors or singular values of a matrix. Spectral clustering for numerical data has been an intensive area of research during the past decade with major contributions coming from various communities including the genetics, image processing, information retrieval and the parallel computing communities. Researchers often show that their spectral clustering method finds a minimum cut in a weighted undirected graph where the nodes are data objects and the weights associated with the edges reflect the similarity between data objects. Much work focuses on the use of spectral techniques to find a relaxed solution to the K-means clustering problem. The relaxed solution then serves as a starting solution for an iterative procedure that determines the final clusters. Another

approach uses a single eigenvector to partition a similarity graph and compute two clusters. Such methods then apply this procedure recursively to produce finer clusters. This approach allows sparsity of the similarity matrix to be exploited and iterative techniques to be used to compute the eigenvector if the similarity graph is large (See, for example, Drineas et al., 2004). A key issue in spectral clustering, and one that differentiates many of the methods, is the construction of the similarity graph (Von Luxburg, 2006).

A key feature of spectral clustering is that it finds a global optimum and as such clustering methods based on spectral techniques are not prone to the problem of convergence to local minima, which often occurs in coordinate descent methods such as the K-means. Spectral clustering often outperforms other algorithms in terms of clustering accuracy (Cheng et al., 2006). In addition, spectral clustering can be implemented efficiently using standard linear algebra software (Von Luxburg, 2006).

Spectral methods have long been the principal tool for handling high-dimensional data since input data is projected into a lower-dimensional space where all computations/comparisons can be performed. Projecting data into a lower-dimensional space helps in extracting data that distinguishes data objects from each other while filtering out noise in the data.

Latent Semantic Indexing (LSI) and Principal Component Analysis (PCA) are two popular techniques that use spectral methods. As mentioned previously, our algorithms build on the concepts presented in LSI (namely, how the term and document matrices are extracted from the term-document data matrix using Singular Value Decomposition (SVD)) and PCA (extraction of principal components). LSI is a document-indexing

algorithm that uses the Singular Value Decomposition (SVD) of a data matrix to index and retrieve documents (Deerwester et al., 1990; Berry et al., 1995; Skillicorn, 2004). PCA is a technique in multivariate analysis used primarily for data reduction (Joliffee, 2002; Wall et al., 2003; Suhr, 2005).

We start this section by providing some linear algebra background for Singular Value Decomposition and Eigen-decomposition (Section 2.1). Next, in Sections 2.2 and 2.3 we illustrate the use of spectral techniques to solve the clustering problem from two perspectives, namely the graph cut and the sum of the squares of error cost function (SSE). The spectral-based algorithms mainly fall into two classes: Hierarchical and Partitional. We discuss some of these algorithms in Section 2.4. We discuss LSI in Section 2.5. PCA is discussed in Section 2.6.

## 2.1 Singular Value Decomposition and Eigen-decomposition

Singular Value Decomposition (SVD) is a mathematical technique for factoring a rectangular matrix into the product of three matrices. Let Y be an m-by-n matrix, then the SVD decomposition of matrix Y is written as

$$Y = USV^t$$

where U is an orthogonal m-by-m matrix, S is an m-by-n diagonal matrix and $V^t$ is an orthogonal n-by-n matrix. The columns of U are called the left singular vectors of Y whereas the columns of V are called the right singular vectors of Y. Both U and V are orthogonal matrices (i.e. $U^tU = I$ and $V^tV = I$, where I is the identity matrix.) The diagonal elements of S, the singular values, are usually arranged in a descending order.

An approximation of matrix Y can be computed using the first k columns of matrices U, S, and V. These k columns correspond to the largest k singular values. Matrix $Y_k$ can be written as

$$Y_k = U_k S_k V_k^t$$

where $U_k$ is an m-by-k matrix of the first k left singular vectors, $V_k$ is an n-by-k matrix of the first k right singular vectors, and $S_k$ is a k-by-k diagonal matrix of the first (largest) k singular values. The Eckart-Young Theorem (Eckart and Young, 1936) states that $Y_k$ is the best rank-k matrix approximation of Y in the Frobenius-norm (matrix 2-norm), in other words, $Y_k$ provides the best k-rank least squares approximation of Y.

From the SVD decomposition of Y, we have the following mathematical relationships among matrices $U_k$, $S_k$ and $V_k$ ,

$$Y_k V_k = U_k S_k$$

and

$$Y_k^t U_k = V_k S_k .$$

If the SVD decomposition of Y is written as $Y = USV^t$, matrix U contains the eigenvectors of $YY^t$ and matrix V contains the eigenvectors of $Y^tY$ (Gram Matrix). Matrix $S^2$ is a diagonal matrix of eigenvalues, which are also the squares of the singular values of Y. We note that

$$YY^t = US^2U^t ,$$

hence

$$(YY^t)U = US^2 ;$$

and

$$Y^t Y = VS^2 V^t,$$

hence

$$(Y^t Y)V = VS^2 \ .$$

## 2.2 Minimizing Graph Cut

The clustering problem can be defined in terms of partitioning a graph. One approach to partitioning the graph is to find the minimum cut while balancing the size of the partitions. The cost of the cut of a graph is usually a function of the weights or number of the edges crossed by a graph cut. In other words, the goal of partitioning a graph is to separate its nodes into groups where there are few edges (or edges with low weights) between nodes belonging to different groups. For more discussion regarding graph theory and graph partitioning, please see Balakrishnan and Ranganathan, 2000, and Boppana, 1987. For the case of 2-way partitioning, this method is more formally known as the graph bisection problem which is NP-hard (Garey et. al., 1976, Wagner and Wagner, 1993).

First, we introduce notation and graph concepts that will be used in this section.

- Let G=(V,E) be an undirected graph where V is the set of vertices $\{v_1,\ldots,v_n\}$ and E is the set of edges that connect any two vertices.
- Each edge in the graph G is assigned a nonnegative weight based on the association between the vertices connected by the edge.

- The matrix W is the adjacency matrix of the graph G where the component at the $i^{th}$ row and $j^{th}$ column, denoted $w_{i,j}$, is the weight of the edge between vertices $v_i$ and $v_j$.
- The matrix D is an n-by-n diagonal matrix whose non-zero elements are the degrees of each vertex in the graph where $d_{i,i}$ (or simply $d_i$) is the degree of vertex $v_i$. The degree of vertex $v_i$ is defined as follows:

$$d_i = \sum_{j=1}^{n} w_{i,j}$$

- The Laplacian matrix L for the graph is defined as L = D – W.
- Let e be an n-dimensional vector whose components are all 1.

For our discussion, each data object is represented by a vertex (node) in the graph. One of the properties of the Laplacian matrix is that for any n-dimensional vector x, the following is defined:

$$x^t Lx = \sum_{i,j=1}^{n} w_{i,j}\left(x_i - x_j\right)^2$$

The cost of the cut of a graph is the sum of the weights of the edges that cross the cut. To simplify the computation, let's assume that the weights of the edges are either 1 or 0. A weight of 1 is assigned to $w_{i,j}$ if there is an edge between $v_i$ and $v_j$, and 0 otherwise. Without loss of generality, we only discuss the case of 2-way partitioning. Let x be a cluster indicator vector where an entry in x, say $x_i$, is set to 1 if the vertex belongs to partition P1 and to -1 if the vertex belongs to P2. Finding the minimum cut of the graph can be formulated as finding the minimum of

$$\sum_{i,j \in E}\left(x_i - x_j\right)^2$$

For all vertices that belong to the same cluster the value of $(x_i-x_j)$ will be zero. Therefore, the above sum calculates the cost of the cut. Given our assumptions about the weights and the indicator vector, the value of the above sum is 4 times the number of edges between the two partitions (Holzrichter and Oliveira, 1999). From the properties of the Laplacian matrix of the graph, it follows that the above sum is equal to $x^tLx$. The partitioning problem can now be stated as minimizing $x^tLx$ subject to the following constraints:

1. $e^tx=0$, that is the vector x is orthogonal to the constant vector e. This condition is to ensure that the partitions are of equal size.
2. $||x||=n^{1/2}$, that is each vertex (data object) may belong to only one partition. ||x|| is norm of the vector x.
3. $x_i \in \{1,-1\}$.

If the third condition is relaxed by allowing x to take on continuous values, the Rayleigh-ritz theorem (Lutkepohl, 1997) shows the above constrained minimization problem can be solved by finding the eigenvector of L associated with the second smallest eigenvalue. Holzrichter and Oliveira (1999) present a similar derivation using Lagrange multipliers. For k-way clustering problems, the Ratio Cut (Hagen and Kahng, 1992) and the Normalized Cut (Shi and Malik, 2000) are used instead of the graph bisection. The Ratio Cut measures the cut cost in relation to the number of vertices in each partition; the normalized cut of a graph partition is the ratio of the cut cost to the total edge connections between the nodes in the partition and all other nodes in the graph. (Von Luxburg, 2006) contains a good illustration of how minimizing the ratio cut and normalized cut lead to a spectral solution for 2-way and k-way clustering.

## 2.3 Minimizing Sum of the Squares of Error function (SSE)

Zha et al. (2001) show that the minimization of the sum of the squares of error (SSE) function leads to a spectral solution associated with the maximization of the trace of the data objects Gram matrix (please see definition below). Given a cluster, SSE measures the deviation of each data object from the associated cluster center. By allowing the cluster indicator vectors to take on arbitrary values, the eigenvectors of the Gram matrix form the solution for the continuous clustering problem. Note that the Gram matrix contains the Euclidean inner product similarity between the data object vectors. To illustrate the work of Zha et al. (2001), we will use the following definitions:

- Let A be an m-by-n data matrix where each data object is a column vector of length m. Let $A_i$ be an m-by-$n_i$ matrix of the of the data objects that belong to the $i^{th}$ cluster. $n_i$ is the number of data objects in the $i^{th}$ cluster.
- Let X be an n-by-k orthonormal matrix of k cluster indicator vectors $x_i = (x_{1,i},\ldots, x_{n,i})^t$ and $x_{i,j}$ is defined as follows:

$$x_{j,i} = n_i / \sqrt{n_i} \quad \text{if} \quad j \in A_i \text{ and 0 otherwise}$$

- $A^tA$ is the Gram matrix which is an n-by-n matrix.
- Let $U_k$ be an n-by-k matrix of the k eigenvectors associated with the k largest eigenvalues of the Gram Matrix.

Zha et al. (2001) show that the SSE function can be written as

$$\text{Trace}(A^tA) - \text{trace}\ (X^tA^tAX)$$

Furthermore, minimizing the SSE function becomes equivalent to maximizing the trace of $(X^tA^tAX)$. By relaxing the constraint on the structure of X and only requiring that X be an orthonormal matrix, the relaxed maximization problem can be solved using the Ky

Fan theorem (Zha et al.,2001). It follows that the optimal value for X is $U_kQ$ where Q is an arbitrary orthogonal matrix. Furthermore, Zha et al. (2001) give a lower bound on the value of SSE in terms of the singular values of the data objects matrix.

## 2.4 Spectral Clustering Algorithms

There are various clustering algorithms that use spectral methods. In Sections 2.2 and 2.3, we discussed two theoretical perspectives for solving the clustering problem using spectral techniques. In this section, we examine three widely referenced algorithms that are based on these theoretical bases.

Shi and Malik (2000) proposed a spectral-based algorithm for image segmentation using a graph-theoretic criterion for measuring the quality of the partitioning. They proposed the normalized cut instead of just the simple graph cut. The normalized cut measures the degree of separation between the clusters as well as the total similarity within the clusters. The normalized cut of a graph partition is the ratio of the cut cost to the total edge connections between the nodes in the partition and all other nodes in the graph. The authors show that minimizing this graph-theoretic criterion leads to a generalized eigenvalue problem. Using the Laplacian matrix of the graph, the authors show that the graph partitioning problem is equivalent to finding the second smallest eigenvector of the of the Laplacian matrix of the graph. Note that each vertex in the graph represents a data object and an edge between two vertices reflects the degree of association between the vertices. The second smallest eigenvector is used to bi-partition the graph. The spectral process can be used recursively to bi-partition each partition, if desired. The authors also proposed an algorithm that would use the top (smallest) j

eigenvectors to represent each data object in j-dimensional space. An algorithm like the K-means can then be used to cluster the j-dimensional vectors into k clusters.

Ng et al. (2001) presents an algorithm that uses the k largest eigenvectors (eigenvectors associated with the k largest eigenvalues) of $D^{-1/2}AD^{-1/2}$ where A is an adjacency matrix of a similarity graph of the data objects and D is a diagonal matrix whose non-zero elements contain the degree of each vertex in the graph. Let $x_i$ and $x_j$ be data objects, then $A_{i,j}$ is defined as follows:

$$A_{i,j} = \exp\left(\frac{-\| x_i - x_j \|^2}{2\sigma^2}\right)$$

$\|..\|^2$ is the square of the distance between any two data objects and $\sigma^2$ is a scaling factor. Let V be a matrix of the k largest eigenvectors where the eigenvectors are the columns of V. The algorithm proceeds by normalizing each of the rows of matrix V to a unit length (each row represents a data object). In short, the algorithm projects the data objects into a new space by computing the k largest eigenvectors of the adjacency matrix; the K-means algorithm is then applied on the projected data points. Ng et al. (2001) explain that mapping the data points into the new space produces tight clusters (more separable) and therefore easier to detect using the popular K-means algorithm. Ng et al. (2001) recommend choosing k equal to the expected number of clusters.

Ding and He (2004) show that principle components are a relaxed solution of the cluster membership indicators in K-means clustering algorithm. Using eigenvectors of the attributes covariance matrix, Ding and He (2004) derived the principle components for the data objects, which are then used to calculate the connectivity between any two data objects. For 2-way clustering, Ding and He (2004) prove that the continuous

solution of the cluster indicator vector is the eigenvector associated with the largest eigenvalue of the Gram Matrix (mean-adjusted data objects similarity matrix). Using SVD terminology, the first right singular vector (singular vector associated with the largest singular value) is the cluster indicator for a 2-way clustering. As an initial approximation of the clustering, Ding and He (2004) propose clustering the data into 2 clusters based on the sign of the values in the singular vector. The K-means algorithm is then used to cluster the data into two clusters (sometimes referred to as the PCA-guided K-means algorithm). For k-way clustering, the authors construct a "PCA based connectivity matrix" which they explain contains the connectivity between the data objects. Let $V_{K-1}$ be a matrix of the first right singular vectors of an m-by-n mean-adjusted data matrix where the data objects are the columns of matrix. The matrix $V_{k-1}V_{k-1}^t$ becomes the PCA connectivity matrix once all negative values and values below a pre-defined threshold are set to zero. The K-means algorithm is then applied to the rows of matrix $V_{k-1}V_{k-1}^t$ to cluster the data into k clusters. The authors suggest that the number of eigenvectors (right singular vectors) should be one less than the number of desired clusters.

Currently, spectral-based clustering algorithms are designed for numeric data where distance or similarity measures between data objects can be calculated. Also, eigenvector decomposition is always applied to the data objects similarity matrix or a graph Laplacian matrix (where each node in the graph represents a data object) which can be computationally expensive to compute.

### 2.4.1 Choosing k (Number of eigenvectors)

How to determine k (number of eigenvectors to use) is currently an open problem with a number of proposed heuristics, none of which has been widely accepted. For k-way spectral-based algorithms, researchers recommend using k (Ng et al., 2001) or k-1 (Ding and He, 2004) eigenvectors to find k clusters where k is the number of clusters. If the number of clusters is unknown, however, as is the case in most clustering tasks, how can the value of k be determined?

"Cluster Validation problem" is an area of research where the goal is to find or validate the "true" number of clusters in a data set. Several methods have been proposed for finding the "true number of groupings" in a data set. For spectral-based algorithms, eigengap heuristics have been proposed to determine the number of eigenvectors (cluster) (Von Luxburg, 2006). (Frayley and Raftery, 1998) propose using Bayes factors to compare different models and choose the best model. (Sugar and James, 2003) propose an information-theoretic approach known as the "jump method" that measures the distortion for each dimension of each observation from the closest cluster center by executing the K-means algorithm for different values of k. Other approaches are gap statistic (Tibshirani et al., 2001) and stability approaches (Ben-david et al., 2006). Nonetheless, determining the proper value of k (number of eigenvectors) is still an open question.

## 2.5 Latent Semantic Indexing

Our approach to categorical clustering using spectral techniques is motivated by Latent Semantic analysis (LSI) used in document retrieval (Deerwester et al., 1990; Berry et al., 1995). The close connection between categorical clustering and the term document

clustering in Information Retrieval (IR) to date has not been exploited by the data mining community. We believe that methods and concepts used in information retrieval can be applied to clustering categorical data as well. In this section, we will use terminology commonly used in Information Retrieval (IR); we use "terms" to refer to attributes and "documents" to refer to data objects.

LSI is a method for automatic indexing and retrieval of documents. LSI encompasses techniques based on frequency and similarity analysis. It is based on the truncated Singular Value Decomposition (SVD) (Golub and Van Loan, 1996) of the term-document matrix (two-mode factor analysis) and is noted for its empirical success in finding latent information in term and document structures that otherwise is not discovered by other IR methods. LSI uses SVD to factor the term-document matrix into linearly independent vectors to extract associations among terms and documents. LSI addresses two main issues, namely polysemy and synonymy, which are hard to address using lexical-based information retrieval methods (Deerwester et al., 1990). Polysemy refers to the fact that certain English words can have multiple meanings (i.e. mean to refer to wicked or the arithmetic mean). Synonymy refers to the fact that several words can be used to refer to the same thing (i.e., automobile and car). These two issues make the retrieval of documents based on term matching ineffective, as the desired documents may not contain the search term. Polysemy and synonymy can be detected by analyzing the relationships and similarities among terms. In LSI, the SVD is used to decompose the term-document matrix into two orthonormal matrices that capture information concerning the terms and documents. These components are used in LSI in a reduced space to extract pertinent information and indirect relationships among the terms and documents.

Each document and term can now be approximated by values in a lower dimensional space than the input space. The LSI methodology of approximating the term vectors by rows of the matrix of the first k left singular vectors (left singular vectors associated with the largest k singular values) motivates our algorithms, which use the rows of the scaled left singular vectors as candidate cluster representatives.

To illustrate how LSI works, let Y be a term-by-document matrix of size m-by-n. SVD decomposes Y into matrices U, S, and V as explained in Section 2.1. As shown in Figure 1, vectors in matrix U are considered the term vectors, and vectors in matrix V are considered the document vectors. The goal of LSI is to capture the most important associations between terms and documents and it accomplishes this task by using the top k terms and documents vectors that are associated with the largest k singular values.

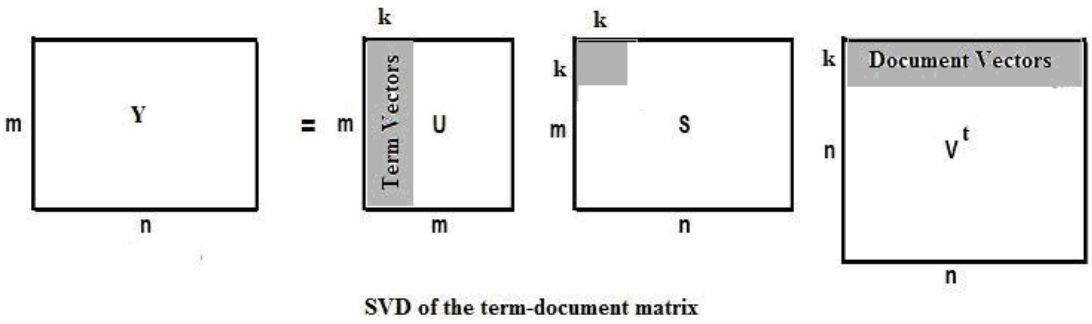


**Figure 4-1 - Singular Value Decomposition**

LSI methods form an approximation matrix $Y_k$ for Y using the top k vectors of U, S and V. As Figure 4-2 shows, $Y_k$ is of rank k since it's in the span of the rows of $V_k^t$.

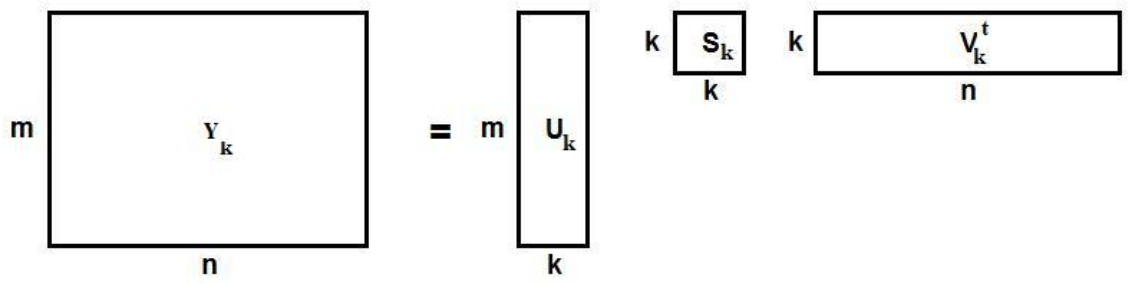


**Figure 4-2 - Approximated term-document matrix**

Since $Y_k$ is an approximation of Y, one can find the similarity between any two terms, documents, or terms and documents using $Y_k$ instead. The terms vectors and document vectors are the coordinates of the points representing the terms and documents, respectively, in a k-dimensional space. To compare two terms, LSI methods typically compute the dot product (similarity) of the two rows of $Y_k$ that correspond to the two terms. The square matrix $Y_k Y_k^t$ contains the dot product (similarity) of every pair of terms. Alternatively, these methods compute the similarity between any two terms using the dot product between two rows of $U_k S_k$ since the square matrix $Y_k Y_k^t$ can be computed using $U_k S_k$ as follows:

$$Y_k Y_k^t = U_k S_k V_k^t \left(U_k S_k V_k^t\right)^t$$
$$Y_k Y_k^t = U_k S_k^2 U_k^t$$

The similarity between two documents can be computed using the dot product between corresponding columns of $Y_k$ or alternatively the columns of $V_k^t S_k$ as shown bellow:

$$= \left(U_k S_k V_k^t\right)^t U_k S_k V_k^t$$
$$= V_k S_k^2 V_k^t$$

Each cell of $Y_k$ contains the similarity between the corresponding term and document of that cell. For example, the cell $y_{i,j}$ contains the similarity between the $i^{th}$ term and the $j^{th}$ document. Alternatively, the similarity between terms and documents can be computed using the rows of matrix $U_k S_k^{1/2}$ and columns of $V_k^t S_k^{1/2}$.

To find the documents that are similar to a given query, the query must be projected into the space as the documents and the terms. In LSI terms, a query is referred to as a pseudo-document. Given q, a vector of query terms, $\hat{q}$ is computed as follows:

$$\hat{q} = q^t U_k S_k^{-1}$$

The cosine similarities between every document (columns of $V_k^t S_k$) and $\hat{q}$ are computed and only those documents whose cosine similarity exceeds a certain threshold are considered to match the query.

Kontostathis and Pottenger (2006) provide a mathematical proof that LSI algorithm (through Singular Value Decomposition (SVD)) captures higher orders of term relationships (transitive relationship). This information is captured in the truncated attribute-to-attribute co-occurrence matrix (Kontostathis and Pottenger, 2002; Kontostathis and Pottenger,2006).

## 2.6 Principal Component Analysis

Principal Component Analysis (PCA) is a non-parametric statistical method that is primarily used in exploratory data analysis as a dimension reduction technique. Karl Pearson introduced PCA in 1901 and it was later developed by Hotelling in

1933(Hotelling, 1933; Jolliffe, 2002). PCA is also known as the Hotelling transform (Hotelling, 1933) or proper orthogonal decomposition (POD) (Lumley, 1967). Our clustering algorithms build on PCA concepts of dimension reduction and the assumption that there is a smaller set of dimensions (attributes) that determine the given set of dimensions (attributes).

PCA takes in a high-dimensional data set of n observation in terms of m attributes (variables) and transforms the data set into a k-dimensional data set where each of the observation is represented with a smaller number of dimensions ($k < m$). The goal of PCA is to reduce the dimensions of the data set by filtering out noise and retaining most of the variations existing in a data set. To illustrate, let Y be an m-by-n matrix of n observations of m attributes (random variables). Each row of Y contains the values for one particular attribute and each column is a vector of the observed values for the different attributes. The goal of PCA is to find the attributes having the maximum variance and to represent each observation in terms of few uncorrelated synthetic attributes. In short, PCA extracts the greatest variations in the data set by utilizing the first k eigenvectors associated with the largest k eigenvalues. For more discussion regarding PCA, see Jolliffe (2002).

**Computations of PCA**

The process of finding the principal components starts by calculating the variance for the observations and ends by calculating the scores for each observation. The steps of PCA can be summarized as follows:

1. Calculate the attribute covariance matrix.
2. Calculate the eigenvectors of the covariance matrix.

3. Choose k eigenvectors that capture as much variations in the data set as possible. k is usually much smaller than m.
4. Calculate the scores for each observation by projecting each mean-adjusted observation into the k-dimension space defined by the k eigenvectors.

**Calculate the covariance matrix**

The covariance matrix can be calculated by subtracting the mean observation from each observation and then dividing by the n. Let Z be the matrix of the mean-adjusted observations where each column of Z is calculated as follows: (note we use superscript to represent matrix columns and subscripts to represent matrix rows)

$$Z^i = Y^i - \overline{Y}$$

and

$$\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y^i$$

The covariance matrix can be calculated as $\frac{ZZ^t}{n}$, which is an m-by-m matrix. The factor 1/(n-1) can be ignored since it is a constant for all elements of the matrix and $ZZ^t$ can be used instead.

**Calculate Eignevectors**

PCA is an orthogonal linear transformation of the data to a new space defined by orthogonal axes (eigenvectors). The eigenvectors of the covariance matrix define the principal directions and are ordered in descending order of the eigenvalues associated with the eigenvectors. The first eigenvector (eigenvector corresponding to the largest eigenvalue) defines the direction of the most variation and the second eigenvector defines the direction with next largest variation and so on. The objective is to choose the

number of eigenvectors that capture most the variation and minimize any information loss. The eigenvectors of $ZZ^t$ are related to the singular value decomposition of Z by the expression,

$$ZZ^tU = US^2,$$

where U is an m-by-m orthogonal matrix ($UU^t = I$) and $S^2$ is an m-by-m diagonal matrix of eigenvalues. S is the diagonal matrix of the singular values of Z.

**Choose k**

The parameter k is chosen so that the maximum variance in the data set in the transformed space is captured. Based on the definitions underlying PCA, the eigenvalues are the variances of the new synthetic variables (principal components). Accordingly, k is chosen so that the ratio of the cumulative sum of the first k eigenvalues to the total sum of the eigenvalues exceeds a specific threshold defined based on the data set and application at hand.

**Project the observation into the new space**

The last step is to project the mean-adjusted observations into the new space defined by the top k eigenvectors as follows:

$$\hat{Z}_k = U_k^t Z$$

where each column of $\hat{Z}_k$ represents an observation in the new space. Also, note that each row of U represents each of the attributes in the new space.

## *3 SCCADDS*

In this section, we present two novel spectral methods for categorical clustering, SCCADDS1 and SCCADDS2. We explore the theoretical framework for the algorithms

and establish their connection to LSI and PCA. We discuss metrics for evaluating the quality of the clusterings. We also present experimental results on standard test data sets that illustrate the quality of the clusters produced and the scalability of the methods.

## 3.1 Overview

The main steps in SCCADDS1 and SCCADDS2 are the following:

1. Create a set of candidate cluster representatives.
2. Project candidate cluster representatives and data objects into a reduced space
3. Evaluate the objective function for each data object.
4. Assign each data object to a cluster defined by its representative.
5. (Optional) Merge the clusters to control granularity of the clustering.

As in CATS, the algorithm presented in Chapter 3, SCCADDS 1 and 2 cluster data objects based on their proximity to candidate cluster representatives which are computed based on data summaries derived from the relationship between attribute categories. Unlike CATS, SCCADDS1 and SCCADDS2 use spectral techniques to determine cluster candidates representatives. The difference between SCCADDS1 and SCCADDS2 lies in 1) how the similarity (proximity) between pairs of attribute categories is defined; 2) the definition of the objective function used to calculate the similarity between each data object and a candidate cluster representative, which determines cluster assignments. SCCADDS1 derives candidate cluster representatives using the attributes cosine similarity matrix and SCCADDS2 derives them using the attributes covariance matrix. (Note, in the following discussion, the term “attribute” and “attribute value” refer to the same thing, namely an attribute category).

## 3.2 Notation

We present the algorithms using "binary data matrices" for illustration purposes and to show the relationships between the different components of the algorithms such as the data matrix, normalized and mean-adjusted data matrices, singular values, eigenvalues, and eigenvectors. In a binary data format, each data object is represented by a vector where each dimension in the vector represents an attribute category. Each dimension in the vector will take on the value 1 if the category associated with it is present in the data object and 0 otherwise. Thereafter, we will refer to these dimensions as attributes. The algorithms can be implemented without materializing the entire binary data matrix and keeping it in storage. For example, the attribute cosine matrix and attribute covariance matrix can be computed using attributes occurrence and co-occurrence frequency vectors. These frequency vectors can be computed efficiently and will occupy less space than a binary data matrix. The following notation will be used to describe the algorithms.

- Let X be an m-by-n binary data matrix where m is the number of attributes and n is the number of data objects. Let $X_i$ denote the $i^{th}$ row in matrix X.
- Let Y be the normalized or mean-adjusted data matrix.

  **For SCCADDS1**, Y is the normalized binary data matrix whose i-th row is

$$Y_i = \frac{X_i}{\| X_i \|}$$

  **For SCCADDS2**, we need to calculate the covariance matrix and as such Y will be the mean-adjusted data matrix. Since we are using a binary data representation, we will illustrate how to calculate the covariance matrix using matrix multiplication. But first, we will digress to present the calculation of the covariance between any

two random variables (attributes) $Z_1$ and $Z_2$. The covariance of any two attributes $Z_1$ and $Z_2$ is computed as follows:

$$\frac{\sum_{i=1}^{n}(z_{1,i} - \overline{z_1})(z_{2,i} - \overline{z_2})}{n}$$

Where n is the number of observations (data objects) and $\overline{z_1}, \overline{z_2}$ are the means of the values for the random variables $Z_1$ and $Z_2$, respectively.

To prepare matrix X for the calculation of the covariance, each row of the matrix is normalized as in SCCADDS1. Then, each normalized row is mean-adjusted by subtracting the mean of each row. Assuming $X_i$ is normalized, each row of Y ($Y_i$ ) is defined as

$$Y_i = X_i - \frac{1}{n}\sum_{j=1}^{n} X_{i,j}$$

- Let $U_k$ be an m-by-k matrix of the first k eigenvectors associated with the k largest eigenvalues of $YY^t$ and $S_k^2$ be a k-by-k diagonal matrix where the diagonal elements contain the eigenvalues values of $YY^t$. Since eigenvalues are the squares of their corresponding singular values, it follows that $S_k$ is a k-by-k diagonal matrix of the singular values of Y.
- Note that $Y_k=U_kS_kV_k$ where $U_k$ is a matrix of the first k left singular vectors of Y, and $V_k$ is a matrix of the first k right singular vectors of Y.

## 3.3 Quality Metrics

In order to determine the clustering quality of SCCADDS, we use the cosine similarity metric to measure the intra-cluster and inter-cluster similarity. For each cluster, we calculate the intra-cluster similarity by taking the average cosine similarity between the data objects and the associated cluster center. The cluster center is simply the average of the values in the data objects vectors. For inter-cluster similarity, we calculate the weighted average of the inter-cluster similarity taking into account the size of each cluster. For a pair of clusters, the inter-cluster similarity is the cosine similarity between their cluster centers. These metrics illustrate how compact and separated the clusters are. The inter-cluster similarity metric can have a value between 0 and 1 with a value closer to 0 indicating that the clusters are well separated. Similarly, the intra-cluster similarity metric can have a value between 0 and 1 with a value closer to 1 indicating high similarity between the data objects in the cluster. In general, the objective is to have high intra-cluster similarity and low inter-cluster similarity.

We also measure the percentage of data objects that change clusters between different executions of the same algorithm (SCCADDS1 or SCCADDS2) using a different value for the algorithms' input parameter, namely the number of eigenvectors. A data object is considered to change clusters if the new cluster is not a sub-cluster or a super-cluster of the old cluster. For example, a cluster that splits into two clusters generates a 0% change in cluster membership. Also, the merging of two clusters generates a 0% change in cluster membership.

## 3.4 SCCADDS1

In SCCADDS1, the proximity matrix contains the cosine similarity between any pair of attribute categories. In SCCADDS1, data objects' complements (as defined in chapter 3) are used to calculate the proximity between each candidate cluster representative and each data object. The use of data objects complements provides a mechanism for assigning negative weights to attribute categories not present in a data object instead of just assigning a weight of zero.

The objective function, that SCCADDS1 maximizes, computes the difference between the cosine of the angle between a cluster representative and a data object and the cosine of the cluster representative and the complement of the data object. The objective is to assign each data object to the cluster representative that is closest to it and farthest from to its complement.

### 3.4.1 Detailed Description

**Step 1: Compute the attribute categories similarity matrix**

We calculate matrix $YY^t$ which is the attributes categories similarity matrix. This matrix contains the cosine measure for each pair of attribute categories. The matrix is m-by-m where m is the total number of attribute categories.

**Step 2: Calculate the eigenvectors**

In this step, we find the first k eigenvectors of the attributes similarity matrix $YY^t$.

**Step 3: Select k eigenvectors**

A value of k is chosen (see Section 3.9).

**Step 4: Project the clusters' representatives into the new space**

Matrix $\Omega_k$, is calculated based on the first k eigenvectors.

$$\Omega_k = (YY^t)U_k$$

Which is equivalent to

$$\Omega_k = U_k S_k^2$$

**Step 5: Project each data object using the first k eigenvectors**

In this step, each data object (columns of matrix X) is projected into the new space using the first k eigenvectors as the axes. Let $\Lambda_k$ represent the projected data objects, we have

$$\Lambda_k = X^t U_k$$

where $\Lambda_k$ is an n-by-k matrix. The rows of $\Lambda_k$ are data objects projected in the new reduced space.

**Step 6: Create an orthogonal binary vector for each data object (a data object complement)**

We create a binary vector for each data object that is orthogonal to the data object vector (the new vector has 1's where there are zeros in the data object binary vector and 0's otherwise). Let matrix B be an n-by-k matrix of the projected data objects complements vectors, and $X_c$ be a matrix of the data objects complements vectors prior to the projection into the new space. Thus, we have

$$B = X_c^t U_k$$

**Step 7: Compare the projected data object vectors to each row of matrix $\Omega_k$**

In this step, we compare each projected data object and its complement to each row of matrix $\Omega_k$. For each data object, we select the row of matrix $\Omega_k$ where the difference between its similarity to the data object and its similarity to the data objects' complement

is maximized. We use the cosine similarity measure. This row is now considered to be the cluster representative of the cluster where the data object will belong.

**Step 8: (Optional) Group the clusters (Merge)**

In this step, a new cluster representative is computed based on the new grouping of the data objects. The new cluster representative is calculated by taking the average of the data objects in the new space for each cluster. We merge the clusters into a final set of clusters using cluster centers. The merge algorithm is based on inter-cluster similarities computed using the cosine measure between cluster centers; it is the same merge process presented in Section 3.8 in Chapter 3.

### 3.4.2 Experimental Evaluation

We have tested SCCADDS1 on the Soybean, Mushroom and Congressional data sets obtained from UCI machine learning repository web site (Asuncion and Newman, 2007). Please refer to Section 5.3 in Chapter 3 for a description of these data sets. Table 4-1 shows the accuracy results of SCCADDS1 with a threshold of 0.7 and k=4, 2, and 2 with an accuracy level of 100%, 88%, and 89% for the Soybean, Congressional Votes, and Mushroom data sets, respectively. Figure 4-3, Figure 4-4 and Figure 4-5 present the accuracy results for SCCADDS1 using other values of k (number of eigenvectors). Table 4-6, Table 4-7, and Table 4-8 in the appendix contain the supporting data. Note that as we increase the number of eigenvectors used in the algorithm, the accuracy levels do not deviate significantly for the Congressional data set but it reaches 96% for the Mushroom data set.

| Data Set Name | Number of data objects | Number of eigen vectors | Number of Clusters in the data set | Number of Clusters Found by SCCADDS1 | Accuracy |
|---|---|---|---|---|---|
| **Soybean** | 47 | 4 | 4 | 4 | 100% |
| **Congressional Votes** | 434 | 2 | 2 | 2 | 88% |
| **Mushroom** | 8,124 | 2 | 2 | 2 | 89% |

**Table 4-1 – Accuracy Results for SCCADDS1 on Standard Test Data sets**

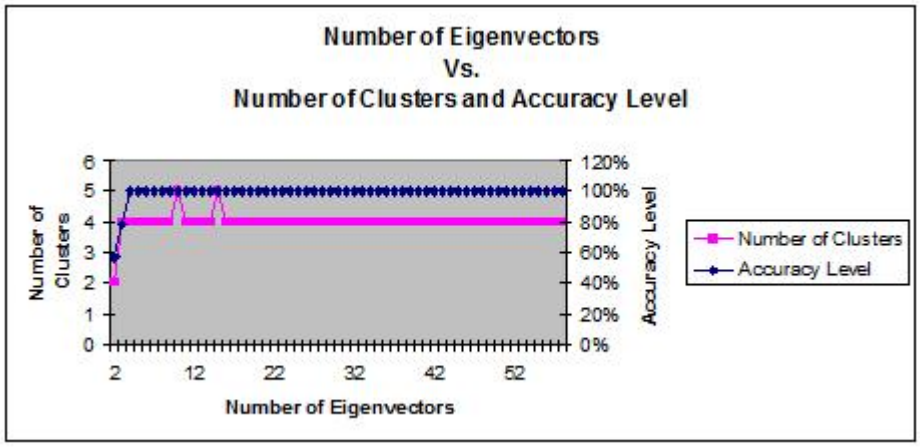


**Figure 4-3 – SCCADDS1 - Number of Eigenvectors VS. Number of Clusters and Accuracy Level for the Soybean Data set**

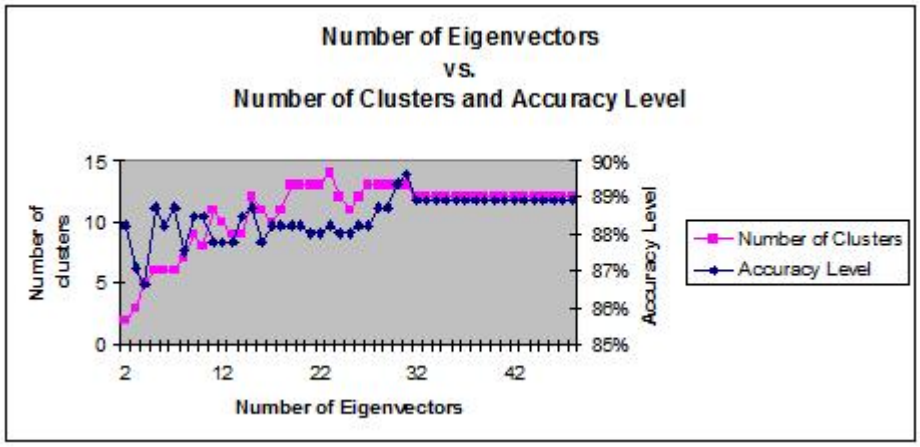


**Figure 4-4 – SCCADDS1 - Number of Eigenvectors VS. Number of Clusters and Accuracy Level for the Congressional Votes Data set**

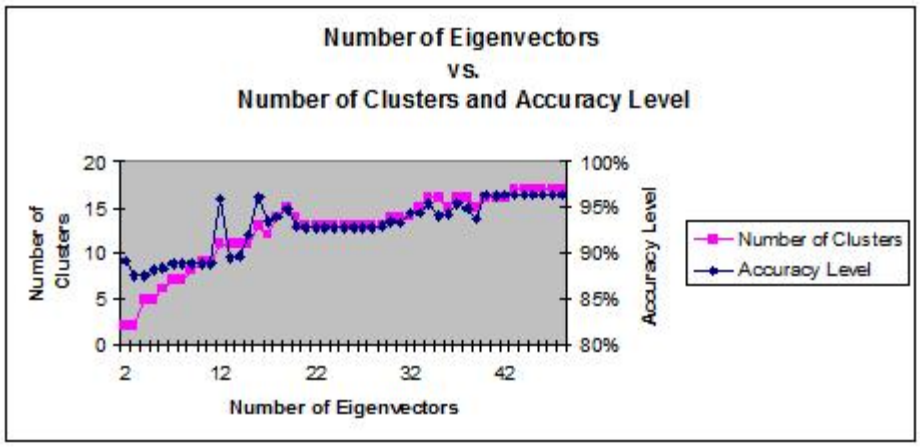


**Figure 4-5 – SCCADDS1 - Number of Eigenvectors vs. Number of Clusters and Accuracy Level for the Mushroom Data set**

## 3.5 SCCADDS2

In SCCADDS2, we use the concepts of PCA to represent the data objects in terms of the principle components and compare them to candidate cluster representatives. As in SCCADDS1, we create an attribute proximity matrix but in SCCADDS2 we use the covariance matrix as it is customarily used in PCA. Ding and He (2004) show that the principal components (eigenvectors of $Y^tY$) are a relaxed solution of the cluster membership indicators in K-means clustering algorithm.

The process of computing the attribute covariance matrix assigns negative weights to attribute categories not present in data objects. The negative weight is equal to the pre-adjustment attribute category average. Since negative weights are used when an attribute category is not present in a data object, there is no need to use a data object's complement in calculating the proximity of a projected data object and a candidate cluster representative. As such, SCCADDS2 uses a dot product to calculate the similarity between a data object and a cluster representative.

### 3.5.1 Detail Description

**Step 1: Compute the covariance attribute matrix**

Computing the covariance is simply the dot product of Y and its transpose. There is no need to divide the entries by n as it is just a scaling factor and will not significantly impact the clustering analysis. The covariance matrix is an m-by-m where m is the number of attribute categories across all attributes. We next remove any rows or columns that contain only zeros. Note that attribute categories that occur in every data object will be removed in this step, as their entries will be 0 once the means is subtracted.

**Step 2: Calculate the eigenvectors**

In this step, we find the first k eigenvectors of the attribute covariance matrix $YY^t$.

**Step 3: Select k eigenvectors**

A value of k is chosen. (Please see Section 3.9.)

**Step 4: Project the clusters representatives into the new space**

Project the clusters representatives into the new space using the first k eigenvectors. We will refer to this matrix as $\Omega_k$. The approximated matrix is calculated as follows:

$$\Omega_k = U_k S_k^2$$

**Step 5: Project each data object vector using the eigenvectors (axes)**

In this step, each data object is projected into the new space using the eigenvectors as axes. Let $\Lambda_k$ represents the projected data objects.

$$\Lambda_k = Y^t U_k$$

where $\Lambda_k$ is an n-by-k matrix. Matrix $\Lambda_k$ contains the scores for each data object.

**Step 6: Compare the projected data objects with each projected cluster representative**

In this step, we compare each projected data object to each candidate cluster representative (rows of matrix $\Omega_k$). For each data object, we select the row with the highest dot product value (similarity). This row represents the cluster of the data object.

**Step 7: (Optional) Group the clusters (Merge)**

This step is the same as step 8 in SCCADDS1.

### 3.5.2 Experimental Evaluation

We have tested SCCADDS2 on the standard test data sets we used for SCCADDS1. SCCADDS2 accuracy levels on these data sets are similar to that produced by SCCADDS1 but using a smaller number of eigenvectors. Table 4-2 shows the accuracy results for SCCADDS2 on the Soybean, Congressional Votes and Mushroom data sets. Accuracy levels of 88% and 89% for the Congressional Votes and Mushroom data sets are achieved using 1 eigenvector in SCCADDS2 versus 2 eigenvectors in SCCADDS1. Figure 4-6, Figure 4-7, and Figure 4-8 show accuracy levels for different executions of SCCADDS2 using a different number of eigenvectors in each execution with merge threshold of 0.7. As in SCCADDS1, the clustering accuracy and the number of clusters increase as more eigenvectors are used in SCCADDS2. For example, the clustering accuracy is 99% for 19 clusters when SCCADDS2 is used to cluster the Mushroom data set using 17 eigenvectors with a merge threshold of 0.5 (Table 4-11).

To illustrate the clustering quality of SCCADDS2 using inter-cluster and intra-cluster similarity, we present the results of multiple executions of SCCADDS2 using different values for k (number of eigenvectors) in Table 4-9, Table 4-10, and Table 4-11 in the appendix. Note, that the ratio of intra-cluster to inter-cluster similarities peaks at

the optimum (published) number of clusters and highest accuracy level for the Soybean and the Congressional data sets in Table 4-9 and Table 4-10.

The percentage of data objects that change cluster membership, in Table 4-9, Table 4-10, and Table 4-11, is very small which indicates that the clusters are splitting as more eigenvectors are used to cluster the data. Note that the change in cluster membership for the Soybean data set is not 0% at levels where the accuracy is 100%; that is because the accuracy is calculated with respect to the 4 published clusters but the percentage of change in cluster membership is calculated using the cluster membership achieved when executing the algorithm with one less eigenvector than what is used in the algorithm when calculating the percentage of change. For example, in Table 4-9, the fifth row shows an 11% change in cluster membership which indicates that some data objects moved from one cluster to another but the change was between two clusters that are part of one published cluster (100% accuracy). Also, note the percentage of change in cluster membership is 0% on the 4$^{th}$ row even though the number of clusters changed from 4 (3$^{rd}$ row) to 5 (4$^{th}$ row); this is an indication of a cluster split.

| Data Set Name | Number of data objects | Number of eigen vectors | Number of Clusters in the data set | Number of Clusters Found by SCCADDS2 | Accuracy |
|---|---|---|---|---|---|
| **Soybean** | 47 | 2 | 4 | 4 | 100% |
| **Congressional Votes** | 434 | 1 | 2 | 2 | 88% |
| **Mushroom** | 8,124 | 1 | 2 | 2 | 89% |

**Table 4-2 - Accuracy Results for SCCADDS2 on Standard Test Data sets**

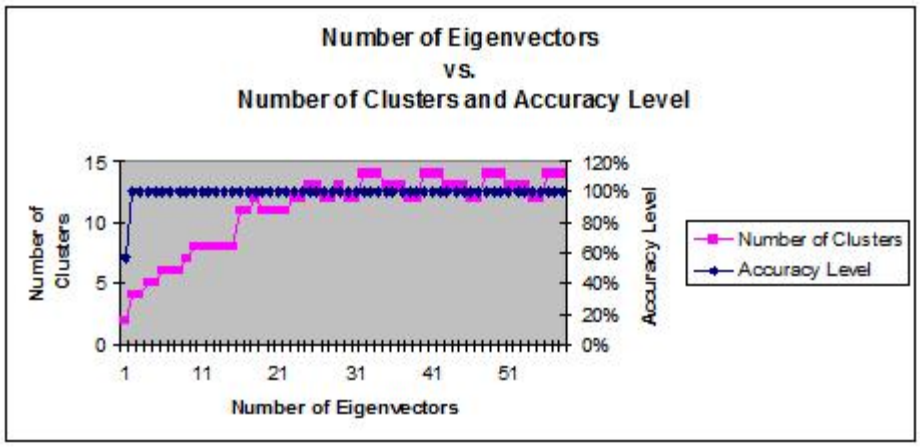


**Figure 4-6 - SCCADDS2 - Number of Eigenvectors VS. Number of Clusters and Accuracy Level for the Soybean Data Set**

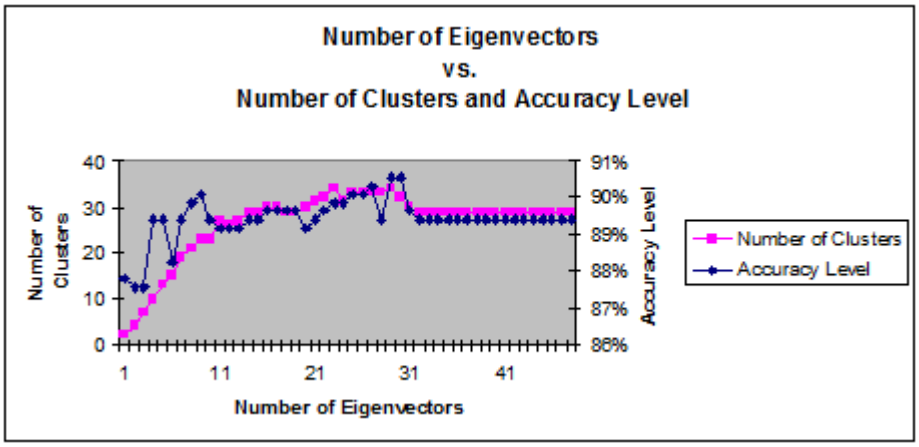


**Figure 4-7 - SCCADDS2 - Number of Eigenvectors VS. Number of Clusters and Accuracy Level for the Congressional Votes Data Set**

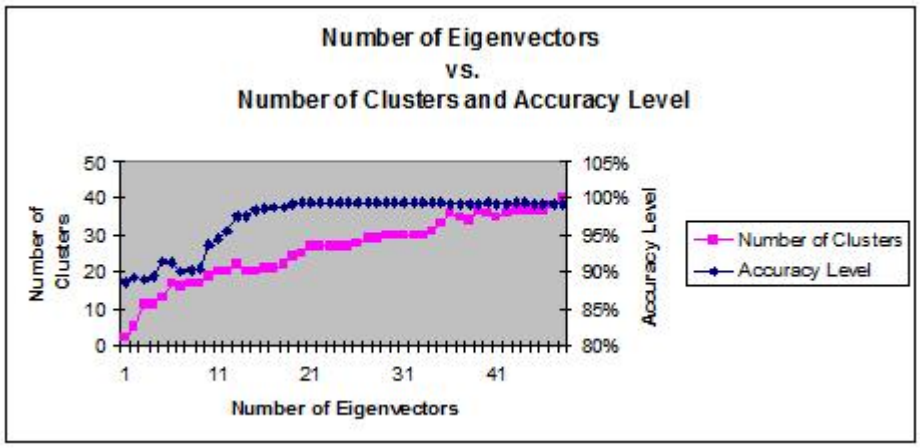


**Figure 4-8 – SCCADDS2 - Number of Eigenvectors VS. Number of Clusters and Accuracy Level for the Mushroom Data Set**

## 3.6 Discussion

The algorithms are motivated by the observation presented in Dhillon (2001) regarding the "Duality of word & document clustering." The observation simply brings to light the premise that documents clusters are induced by terms clusters and vice versa. In our view, the setting of categorical data clustering is identical to that of documents clustering where attribute categories are represented by terms and data objects are represented by documents. As such, following Dhillon's observation, we can use attribute categories clusters to cluster data objects in categorical data sets. A similar concept is used in the methodology of CACTUS (Ganti et al., 1999) and CLICKS (Zaki et al., 2007). The early phases of CLICKS and CACTUS mainly find those sets of attribute categories that are strongly connected based on their occurrence and co-occurrence frequencies in the data objects (please see Chapter 2 for a discussion of CLICKS and CACTUS). These attribute categories sets are then used to form "candidate" clusters' representatives which are later validated by scanning the data objects for the presence of these attribute values. With that point in mind, SCCADDS1 finds the potential cluster candidates by finding attribute clusters using the attribute categories similarity matrix. SCCADDS2 follows the same procedure but uses the covariance matrix instead.

### 3.6.1 Clustering Attributes

To find attribute categories clusters, SCCADDS1 and SCCADDS2 build on the work presented by Zha et al. (2001) where spectral techniques are shown to provide a relaxed solution (global optimum) for minimizing the sum of the squares of error cost function (SSE) between data objects and cluster centers. Since the attribute categories are considered the data objects to be clustered, spectral techniques are applied to the attribute

similarity matrix (SCCADDS1) and attribute covariance matrix (SCCADDS2). As shown in Zha et al.(2001), the first k eigenvectors associated with the k largest singular values of the data objects similarity matrix form the optimal solution for the non-discrete clustering problem (clusters indicator matrix). According to Zha et al. (2001), the solution is $U_kQ$ where Q is an arbitrary orthogonal matrix. Let Q be the identity matrix I, then $U_k$ can be a solution to the problem. We then can project the attribute categories association matrix into the space defined by the cluster indicators vectors ($YY^tU_k = U_kS_k^2$). Each row of matrix $U_kS_k^2$ corresponds to an attribute category and as such it contains the coordinates for each attribute category in the space defined by the columns of $U_k$. Each column of $U_k$ defines a cluster membership indicator vector and each value in that column defines the intensity of the membership of each attribute category in that cluster. It follows that similar attribute categories would have similar values (close coordinates) with respect to each column of $U_k$. Each row of $U_kS_k^2$ is considered a cluster representative since it contains the coordinates for the attribute categories in the new space (each row of the attribute categories association matrix represents a cluster on that dimension where all data objects in the cluster have the same value for that attribute). Note that each row of matrix $YY^t$ contains weights that reflect the relationship between attribute categories pairs. These weights are also based on the co-occurrence of the attribute categories in the data objects and as such implicitly contain the relationships between data objects.

Since it is not possible to directly compare the data objects to these clusters' representatives (columns of $U_k$), SCCADDS algorithms take the approach of projecting

the data objects into the same space ($X^tU_k$ for SCCADDS1 and $Y^tU_k$ for SCCADDS2) as the attribute categories.

### 3.6.2 SCCADDS and LSI

SCCADDS1 and SCCADDS2 build upon the concepts of LSI. In SCCADDS1, we use normalized attribute categories co-occurrence frequencies instead of absolute co-occurrence frequencies as would be used in LSI if a binary vector representation is used. Unlike absolute frequencies, these normalized frequencies provide a weight for each pair of attribute categories which takes into account the frequency of each attribute category in the data set; therefore, a pair of attribute categories that co-occurs very frequently in the data set will have lower weights (less significant) than other pairs of attribute categories that co-occur/occur less often.

In LSI, the matrix $U_kS_k$ contains the term vectors and the matrix $V_kS_k$ contains the document vectors. Each row of $U_kS_k$ represents a term and each row of $V_kS_k$ represents a document. Similarly, in SCCADDS, we calculate matrix $U_kS_k^2$ which is similar to matrix $U_kS_k$ (term matrix) but differing only by a scaling factor of the axes of $S_k$. In LSI, the columns of $U_k$ represent some artificial concepts that are present in a data set. In clustering terminology, these artificial concepts represent clusters representatives. Since $U_kS_k^2 = (YY^t)U_k$, it follows that the rows of $U_kS_k^2$ represent the rows of $YY^t$ projected into a space defined by the orthogonal axes defined by the columns of $U_k$ (clusters representatives). The rows of $YY^t$ correspond to the attribute categories; the rows of $U_kS_k^2$ represent the attribute categories in the new space. Note that in LSI, queries are considered pseudo-documents and as such are projected into the new space

prior to being compared to the data objects. Similarly, SCCADDS1 and SCCADDS2 project the attribute values (rows of $YY^t$) into the new space and then compare them to the data objects.

The algorithms cluster the data objects by comparing the data objects in the new reduced space to the attribute categories in the new space ($U_k S_k^2$). The rows of $U_k S_k^2$ partition the new space based on their similarity to each other. Therefore, all attribute categories that co-occur together will have close coordinates and all data objects that contain these attribute categories will also have close coordinates.

### 3.6.3 SCCADDS and PCA

In SCCADDS2, we find the scores for each data object using the principle components for the data set ($Y^tU_k=V_kS_k$). PCA allows us to represent each data object using a small number of dimensions that capture the most variations in the data set.

Ding and He (2004) showed that the first k eigenvectors that are associated with the k largest eigenvalues of the Gram Matrix ($Y^tY$) form the continuous solution for the cluster indicators problem where Y is the mean-adjusted matrix used in computing the attribute covariance matrix. Using SVD decomposition of $Y=USV^t$, the relaxed solution would be the first k eigenvectors of matrix V. By SVD decomposition of Y, matrix $Y^tU_k=V_kS_k$. Therefore, in SCCADSS2, the rows of the matrix $Y^tU_k$ have two interpretations: 1) (PCA interpretation) they contain the scores for each data object using the first k principal components; 2) (continuous cluster indicators) they are coordinates for the data objects in a space defined by the attributes clusters membership indicators.

## 3.7 Comparative Analysis

In this section, we present comparative results on standard and synthetic data sets.

### 3.7.1 Standard Data sets

We compared the results of SCCADDS1 and SCCADDS2 on the Soybean, Congressional Votes and Mushroom data sets with the results of CLICKS, COOLCAT, LIMBO, ROCK and Eigencluster (Cheng et al., 2006 ). For a description of CLICKS, COOLCAT, LIMBO, and ROCK, please refer to Chapter 2. The Eigencluster is a spectral-based divide-and-merge algorithm that produces hierarchical clusters. Eigencluster has two phases. The divide phase consist of dividing the data set into a hierarchy of clusters ( a tree) and the second phase merges the clusters using any of the well-known objective functions such as sum of the squares of error(SSE), min-sum , etc. The authors emphasize that the merge algorithm should respect the hierarchy (tree) boundaries; that is two nodes can only be merged if they share that same parent node. Furthermore, the merge algorithm uses a user defined threshold that guides the merge process. The authors recommend using the spectral algorithm used in (Kannan et al., 2004) for the divide phase and have provided an implementation on the web that we use in our comparative analysis.

Table 4-3 and Table 4-4 present comparative results for clustering the Congressional Votes and Mushroom data sets into two clusters. Note, SCCADDS algorithms achieve the highest clustering accuracy for the Congressional data set and achieve a clustering accuracy comparable to that of LIMBO for the Mushroom data set.

| | Accuracy | Entropy | Remarks |
|---|---|---|---|
| **SCCADDS1/SCCADDS2** | 88% | 0.452 | This is the result for 2 clusters. Higher accuracy levels may be achieved with a higher number of clusters |
| **Traditional Hierarchical Clustering Algorithm** | 86% | | 157 and 215 records were properly classified (Guha et al., 1999). |
| **ROCK** | 79% | 0.499 | 144 and 201 records were properly classified. 62 records were excluded as they were considered outliers and were not classified (Guha et al., 1999). We use a data set size of 434 when we calculated the accuracy level for ROCK to provide comparability with other algorithms presented.<br>Entropy results were obtained from Cheng et al., 2006. |
| **LIMBO** | 87% | | Results obtained from Andritsos et al., 2004. |
| **COOLCAT** | 85% | 0.498 | Accuracy results are from (Andritsos et al., 2004). Entropy results are from (Cheng et al., 2006). |
| **CLICKS** | Not available for 2 clusters | Not available for 2 clusters | An accuracy level of 96% is achieved with 13 clusters. One of the 13 groups is designated for outliers (Zaki et al., 2007). |
| **Eigencluster** | | .48 | Entropy results were obtained from Cheng et al., 2006. |

**Table 4-3 - Comparative Results for the Congressional Votes Data Set (2 Clusters)**

| | Accuracy | Remarks |
|---|---|---|
| **SCCADDS1/SCCADDS2** | 89% | This is the accuracy level for 2 clusters. Higher accuracy levels may be achieved with a higher number of clusters. |
| **ROCK** | 57% | This is the clustering result with 2 clusters (Andritsos et al., 2004). Guha et al. (1999) implemented Rock that partitioned the data set into 21 clusters and could not further combine the clusters to form the two pre-defined classes; only 32 records were not clustered properly. |
| **LIMBO** | 89% | Results for 2 clusters obtained from Andritsos et al., 2004. |
| **COOLCAT** | 73% | Accuracy results for 2 clusters are from Andritsos et al., 2004. |
| **CLICKS** | Not available for 2 clusters | An accuracy level of 97% is achieved with 19 clusters (Zaki et al., 2007). |
| **Eigencluster** | 81% | Precision and recall (Cheng et al., 2006). |

**Table 4-4 - Comparative Results for the Mushroom Data Set (2 Clusters)**

### 3.7.2 Synthetic Data sets

We tested SCCADDS1 and SCCADDS2 on the same synthetic data sets that we constructed for CATS in Chapter 3 (see Section 5.5.2 for a description of the data sets).

| | | | | SCCADDS1 | | | | SCCADDS2 | | | | Eigencluster | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Set Name | Number of data objects | Noise Ratio | Number of Clusters in the dataset | Number of eigen vectors | Number of clusters found | Accuracy | Remarks | Number of eigen vectors | Number of clusters found | Accuracy | Remarks | Alpha | Number of clusters | Accuracy | Remarks |
| DS1 | 1,000 | 0% | 5 | 5 | 5 | 100% | No change in cluster memebership for values of k >5 | 4 | 5 | 100% | No change in cluster memebership for values of k >4 | 0.4 | 5 | 93% | |
| DS2 | 5,000 | 0% | 10 | 9 | 10 | 100% | No change in cluster memebership for values of k >9 | 9 | 10 | 99% | No change in cluster memebership for values of k >9 | 0.4 | 10 | 91% | |
| DS3 | 5,000 | 2% | 10 | 10 | 10 | 100% | No change in cluster memebership for values of k >10 | 9 | 10 | 100% | No change in cluster memebership for values of k >9 | 0.4 | 10 | 91% | |
| DS4 | 5,000 | 10% | 10 | 10 | 10 | 100% | No change in cluster memebership for values of k >10 | 9 | 10 | 99% | No change in cluster memebership for values of k >9 | 0.4 | 10 | 85% | |
| DS5 | 5,000 | 10% | 10 | 9 | 10 | 100% | No change in cluster memebership for values of k >9 | 7/9 | 10 | 99%/100% | No change in cluster memebership for values of k >9 | 0.2 | 9 | 82% | We tried different values for Alpha but we could not get the algorithm to produce exactly 10 clusters. The algorithm produced 11 clusters with an alpha of .35 and an accuracy of 95% |

**Figure 4-9 - Comparative Results on Synthetic Data Sets**

Figure 4-9 shows comparative analysis for clustering accuracy results for SCCADDS1, SCCADDS2, and Eigencluster (Cheng et al., 2006; Kannan et al., 2004)[5]. SCCADDS1 and SCCADDS2 clearly outperform Eigencluster on these data sets. SCCADDS algorithms find the correct number of clusters with an accuracy of more than 99% for all data sets. Moreover, SCCADDS1 and SCCADDS2 are not as sensitive to noise in data as Eigencluster is. Table 4-12 and Table 4-13 in the appendix present accuracy results, and percentage of change in cluster membership among data objects for multiple executions of SCCADDS1 and SCCADDS2 for different values of k using synthetic data sets (DS1-DS5). Table 4-14 in the appendix presents the intra-cluster and inter-cluster similarities for data set DS5. Note that the ratio of intra-cluster to inter-cluster similarity peaks at the highest accuracy level and optimal number of clusters (pre-defined number of clusters for DS5).

## 3.8 Scalability Analysis

In this section we present the results of scalability testing for SCCADDS1 and SCCADDS2 in terms of the number of data objects, and the number of attributes and attributes categories.

**Number of Data Objects**

We created 10 data sets that range in size from 6,500 to 65,000 data objects to test the scalability of SCCADDS in terms of data set size. Figure 4-10 illustrates that the change in elapse time is linear with the change in the data set size (number of data objects) when SCCADDS2 is executed using these data sets. The scalability testing of SCCADDS1 produces results similar to that of SCCADDS2.

[5] The clustering engine for Eigencluster is provided by the authors of Eigencluster at http://arc2.cc.gatech.edu/cluster html.

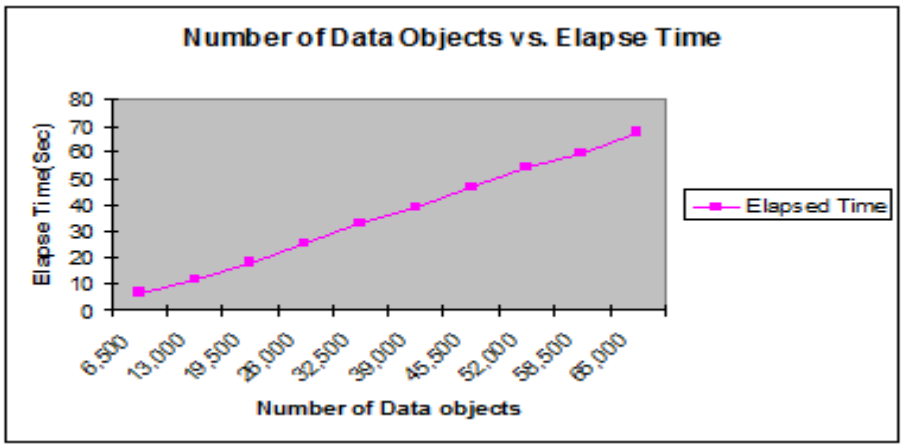


**Figure 4-10 - Scalability testing - Number of data objects**

### Number of Attributes

We used the same data sets we used for testing the scalability of CATS in Chapter 3. We created five synthetic data sets with 5,000 data object in each data set. Each data set has a different number of attributes. The domain size for each attribute is 10. Table 4-5 presents the results of the experiments on these five data sets. Please note that columns 5 and 9 (number of attribute categories) contain the actual number of attribute categories (any categories that do not occur or occur in every data objects have been removed). Figure 4-11 shows how the execution time (elapse time) of SCCADDS2 varies as the number of attribute categories is increased (plot of data from Table 4-5 ).

| | | | | SCCADDS1 | | | | SCCADDS2 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Set Name | Number of Data Objects | Number of Attributes | Domain | Number of Attributes Categories | Elapsed Execution Time (sec.) | % change in Number of attributes | % of Change in Elapsed Execution Time | Number of Attributes Categories | Elapsed Execution Time (sec.) | % change in Number of attributes | % of Change in Elapsed Execution Time |
| DS1A | 5000 | 5 | 10 | 38 | 4.2 | 0 | 0 | 38 | 4.2 | 0 | 0 |
| DS2A | 5000 | 10 | 10 | 94 | 7.1 | 147.37% | 69.05% | 94 | 7 | 147.37% | 66.67% |
| DS3A | 5000 | 15 | 10 | 150 | 11.1 | 59.57% | 56.34% | 150 | 10.9 | 59.57% | 55.71% |
| DS4A | 5000 | 20 | 10 | 200 | 14.8 | 33.33% | 33.33% | 200 | 14.1 | 33.33% | 29.36% |
| DS5A | 5000 | 30 | 10 | 300 | 23.8 | 50.00% | 60.81% | 300 | 22.5 | 50.00% | 59.57% |

**Table 4-5 - Scalability Testing for SCCADDS1 and SCCADDS2 - Number of Attributes**

As the number of attributes increases, the execution times increases by the similar proportion for SCCADDS1 and SCCADDS2. We expect SCCADDS time complexity to be quadratic in terms of the number of attribute categories (see Section 3.10). However, based on our experiments, the change in elapse time is almost linear with the change in the number of attribute categories. The experiments show that SCCADDS is linear in terms of the number of attribute categories because the number of attribute categories is small relative to the number of data objects which is typical in most practical applications. In other words, most of the elapsed time is spent in comparing the data objects to candidate cluster representatives.

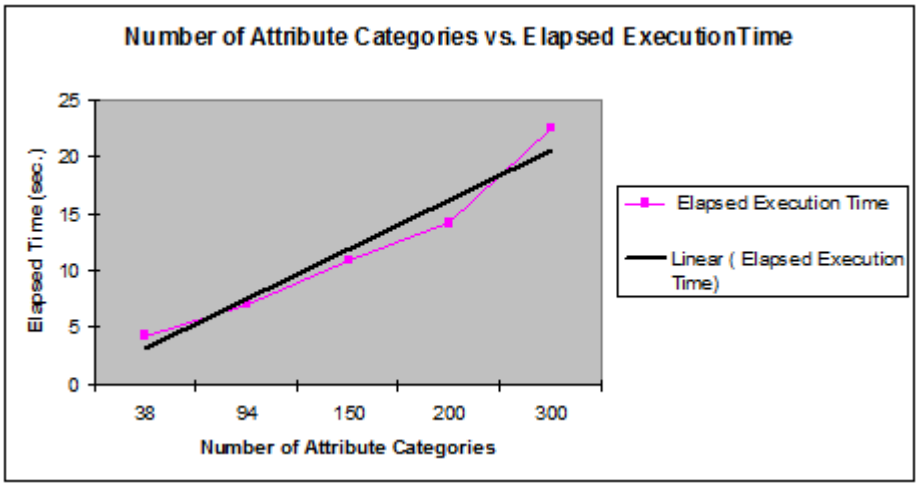


**Figure 4-11 – SCCADDS2 – Scalability Testing**

## 3.9 The Effect of k on the performance of SCCADDS1 and SCCADDS2

As presented in the appendix (Tables 4.6 to 4.11), the accuracy levels for SCCADDS1 and SCCADDS2 do not deviate significantly as the number of eigenvectors changes for different executions of the algorithms. As more eigenvectors (and eigenvalues) are used

in representing the data, more information and more noise are incorporated into the clustering process; this in turn results in more clusters or splits of the clusters already discovered at a lower number of eigenvectors. Table 4-10 and Table 4-11 show the percentage of data objects that change clusters between any two consecutive executions of SCCADDS2 for the Congressional Votes and Mushroom data sets. As shown in these tables, the percentage of change is mostly under 5% and that is an indication that the clusters tend to split as we use more eigenvectors in projecting the data. In short, our empirical results on standard and synthetic data show that SCCADDS1 and SCCADDS2 are not very sensitive to changes in the value of k (number of eigenvectors) once a stable clustering level has been achieved (low percentage of change in cluster membership). In our experiments, the number of eigenvectors that produced the best accuracy level and the correct number of clusters is at most the number of "known" clusters in the data sets, which is in agreement with other researchers' recommendations for choosing k. In the absence of any knowledge of the "true" number of clusters, we propose using a number of eigenvectors where there is very little change of cluster membership among the data objects or where the sum of the squares of error (SSE) is the lowest for the number of clusters desired. In addition, the intra-cluster similarity, inter-cluster similarity, and their ratio are good indicators of cluster quality and can be used to select the appropriate number of eigenvectors.

## 3.10 Complexity Analysis of SCCADDS1 and SCCADDS2

SCCADDS1 and SCCADDS2 consist of mainly four phases. The first phase is for constructing the attribute proximity matrix. The second phase is where the eigenvectors are computed and the data objects are projected into the new space. The data objects are

compared to the candidate cluster representatives in the third phase. The fourth phase, which is optional, is where the cluster centers are merged. For the following discussion, let m be the number of attribute categories, k is the number of eigenvectors, and n be the number of data objects.

**Phase 1 – Constructing the attribute categories proximity matrix.**

The attribute categories proximity matrix can be constructed through one pass through the data by creating counters to hold the attribute categories occurrence and co-occurrence frequencies. This step would have time complexity of $O(n\frac{m^2}{2})$.

**Phase 2 - Computation of the eigenvectors and projection of data objects into the new space**

The time complexity will depend on the method used to find the eigenvectors. Certain efficient methods can be on $O(km^2)$.

**Phase 3 - Cluster Assignments**

In this phase, each projected data object is compared to each candidate cluster. The time complexity for this phase will depend on the number of data objects and the number of candidate clusters O(nm).

**Phase 4 – Merge Algorithm**

The execution time of the merge algorithm depends on the number of candidate clusters assigned data objects, which is significantly lower than m (based on experimental results).

## 3.11 Practical Consideration

As noted earlier, the attribute similarity and attribute covariance matrices can be computed without expanding the data into binary vectors format. The computation of the attribute similarity or covariance matrices of a data set requires one scan of the data set to calculate the frequency of each attribute category and the co-occurrence frequency of each pair of attributes categories. A counter for each attribute category and another counter for each pair of attribute categories are initialized to zero. For each data object in the data set, the appropriate counters are incremented. The cosine similarity of each pair of attributes categories is the co-occurrence frequency divided by the square root of the frequency of each attribute category. To illustrate the computation of the covariance matrix, we define matrices G and D, and vector f. Matrix G is an m-by-m matrix that contains attribute categories co-occurrence frequencies where m is the number of attribute categories. The m-by-m diagonal matrix D and the m-length vector f contain the attribute categories frequencies. The value n is the number of data objects in the data set. The attribute covariance matrix is calculated as follows: $\left(D^{-1/2}GD^{-1/2}\right)-\left(\frac{f^{1/2}(f^{1/2})^t}{n}\right)$.

The steps of SCCADDS can easily be implemented using parallel computing resources. Using parallel computing can significantly cut down on the elapse time of an algorithm making it more practical to implement and use.

## *4 Remarks*

In this chapter, we introduced two novel algorithms that use spectral techniques to cluster categorical data. SCCADDS1 and SCCADDS2 present a new framework for using spectral techniques on categorical data sets. We have discussed the theoretical

framework for SCCADDS and demonstrated, by experimental evaluation on standard and synthetic data sets, the clustering accuracy and performance of SCCADDS. The algorithms are easy to implement and can easily scale to large data sets. Furthermore, the steps of SCCADDS algorithms can be partitioned to run in parallel to reduce clock time of the algorithms which adds the applicability of SCCADDS to handle large data sets.

For future research, we will be extending SCCADDS to implement fuzzy clustering where each data object can belong to more than one cluster. SCCADDS will be enhanced to rank the clusters in terms of their similarity to a data object.

# 5 Appendix to Chapter 4

**Table 4-6 – Accuracy Results for the Soybean data set using SCCADDS1 and different values of k (number of eigenvectors ) – threshold for the merge algorithm is 0.7.**

| Number of Eigenvectors | Number of Clusters | Accuracy |
|---|---|---|
| 2 | 2 | 57% |
| 3 | 4 | 79% |
| 4 | 4 | 100% |
| 5 | 4 | 100% |
| 6 | 4 | 100% |
| 7 | 4 | 100% |
| 8 | 4 | 100% |
| 9 | 4 | 100% |
| 10 | 5 | 100% |
| 11 | 4 | 100% |
| 12 | 4 | 100% |
| 13 | 4 | 100% |
| 14 | 4 | 100% |
| 15 | 5 | 100% |
| 16 | 4 | 100% |
| 17 | 4 | 100% |
| 18 | 4 | 100% |
| 19 | 4 | 100% |
| 20 | 4 | 100% |
| 21 | 4 | 100% |
| 22 | 4 | 100% |
| 23 | 4 | 100% |
| 24 | 4 | 100% |
| 25 | 4 | 100% |
| 26 | 4 | 100% |
| 27 | 4 | 100% |
| 28 | 4 | 100% |
| 29 | 4 | 100% |
| 30 | 4 | 100% |
| 31 | 4 | 100% |
| 32 | 4 | 100% |
| 33 | 4 | 100% |
| 34 | 4 | 100% |
| 35 | 4 | 100% |
| 36 | 4 | 100% |
| 37 | 4 | 100% |
| 38 | 4 | 100% |
| 39 | 4 | 100% |
| 40 | 4 | 100% |
| 41 | 4 | 100% |
| 42 | 4 | 100% |
| 43 | 4 | 100% |
| 44 | 4 | 100% |
| 45 | 4 | 100% |
| 46 | 4 | 100% |
| 47 | 4 | 100% |
| 48 | 4 | 100% |
| 49 | 4 | 100% |
| 50 | 4 | 100% |
| 51 | 4 | 100% |
| 52 | 4 | 100% |
| 53 | 4 | 100% |
| 54 | 4 | 100% |
| 55 | 4 | 100% |
| 56 | 4 | 100% |
| 57 | 4 | 100% |
| 58 | 4 | 100% |

**Table 4-7 -Accuracy Results for the Congressional Votes data set using SCCADDS1 and different values of k (number of eigenvectors ) – threshold for the merge algorithm is 0.7.**

| Number of Eigenvectors | Number of Clusters | Accuracy |
|---|---|---|
| 2 | 2 | 88% |
| 3 | 3 | 87% |
| 4 | 5 | 87% |
| 5 | 6 | 89% |
| 6 | 6 | 88% |
| 7 | 6 | 89% |
| 8 | 7 | 88% |
| 9 | 9 | 88% |
| 10 | 8 | 88% |
| 11 | 11 | 88% |
| 12 | 10 | 88% |
| 13 | 9 | 88% |
| 14 | 9 | 88% |
| 15 | 12 | 89% |
| 16 | 11 | 88% |
| 17 | 10 | 88% |
| 18 | 11 | 88% |
| 19 | 13 | 88% |
| 20 | 13 | 88% |
| 21 | 13 | 88% |
| 22 | 13 | 88% |
| 23 | 14 | 88% |
| 24 | 12 | 88% |
| 25 | 11 | 88% |
| 26 | 12 | 88% |
| 27 | 13 | 88% |
| 28 | 13 | 89% |
| 29 | 13 | 89% |
| 30 | 13 | 89% |
| 31 | 13 | 90% |
| 32 | 12 | 89% |
| 33 | 12 | 89% |
| 34 | 12 | 89% |
| 35 | 12 | 89% |
| 36 | 12 | 89% |
| 37 | 12 | 89% |
| 38 | 12 | 89% |
| 39 | 12 | 89% |
| 40 | 12 | 89% |
| 41 | 12 | 89% |
| 42 | 12 | 89% |
| 43 | 12 | 89% |
| 44 | 12 | 89% |
| 45 | 12 | 89% |
| 46 | 12 | 89% |
| 47 | 12 | 89% |
| 48 | 12 | 89% |

**Table 4-8 -Accuracy Results for the Mushroom data set using SCCADDS1 and different values of k (number of eigenvectors ) – threshold for the merge algorithm is 0.7.**

| Number of Eigenvectors | Number of Clusters | Accuracy |
|---|---|---|
| 2 | 2 | 89% |
| 3 | 2 | 88% |
| 4 | 5 | 88% |
| 5 | 5 | 88% |
| 6 | 6 | 88% |
| 7 | 7 | 89% |
| 8 | 7 | 89% |
| 9 | 8 | 89% |
| 10 | 9 | 89% |
| 11 | 9 | 89% |
| 12 | 11 | 96% |
| 13 | 11 | 90% |
| 14 | 11 | 90% |
| 15 | 11 | 92% |
| 16 | 13 | 96% |
| 17 | 12 | 93% |
| 18 | 14 | 94% |
| 19 | 15 | 95% |
| 20 | 14 | 93% |
| 21 | 13 | 93% |
| 22 | 13 | 93% |
| 23 | 13 | 93% |
| 24 | 13 | 93% |
| 25 | 13 | 93% |
| 26 | 13 | 93% |
| 27 | 13 | 93% |
| 28 | 13 | 93% |
| 29 | 13 | 93% |
| 30 | 14 | 94% |
| 31 | 14 | 93% |
| 32 | 14 | 94% |
| 33 | 15 | 94% |
| 34 | 16 | 95% |
| 35 | 16 | 94% |
| 36 | 15 | 94% |
| 37 | 16 | 95% |
| 38 | 16 | 95% |
| 39 | 15 | 94% |
| 40 | 16 | 96% |
| 41 | 16 | 96% |
| 42 | 16 | 96% |
| 43 | 17 | 96% |
| 44 | 17 | 96% |
| 45 | 17 | 96% |
| 46 | 17 | 96% |
| 47 | 17 | 96% |
| 48 | 17 | 96% |

**Table 4-9- Accuracy Results for the Soybean data set using SCCADDS2 and different values of k (number of eigenvectors ) – threshold for the merge algorithm is 0.5.**

| Number of Eigenvectors | Number of Clusters | Accuracy | SSE | % change in cluster membership | within-Cluster similarity | Inter-cluster similarity | Ratio of intra/ inter similarity |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 57% | 368.063 | 0% | 79% | 58% | 136% |
| 2 | 4 | 100% | 232.8824 | 0% | 87% | 52% | 169% |
| 3 | 4 | 100% | 232.8824 | 0% | 87% | 52% | 169% |
| 4 | 5 | 100% | 222.619 | 0% | 88% | 55% | 159% |
| 5 | 5 | 100% | 218.8333 | 11% | 88% | 56% | 157% |
| 6 | 5 | 100% | 220.8846 | 9% | 88% | 55% | 159% |
| 7 | 5 | 100% | 215.1 | 11% | 88% | 56% | 159% |
| 8 | 6 | 100% | 199.5077 | 2% | 89% | 54% | 165% |
| 9 | 6 | 100% | 202.5667 | 11% | 89% | 54% | 164% |
| 10 | 6 | 100% | 199.5636 | 2% | 89% | 54% | 164% |
| 11 | 7 | 100% | 191.2 | 11% | 90% | 54% | 165% |
| 12 | 6 | 100% | 195.8286 | 13% | 90% | 54% | 165% |
| 13 | 7 | 100% | 183.6286 | 0% | 90% | 56% | 160% |
| 14 | 7 | 100% | 186.9143 | 2% | 90% | 57% | 159% |
| 15 | 7 | 100% | 183.6286 | 2% | 90% | 56% | 160% |
| 16 | 8 | 100% | 180.5203 | 9% | 90% | 56% | 160% |
| 17 | 9 | 100% | 178.187 | 2% | 90% | 56% | 163% |
| 18 | 10 | 100% | 166.6286 | 4% | 91% | 56% | 163% |
| 19 | 8 | 100% | 177.7397 | 0% | 91% | 55% | 163% |
| 20 | 9 | 100% | 166.9833 | 4% | 91% | 57% | 160% |
| 21 | 9 | 100% | 168.8444 | 2% | 91% | 56% | 161% |
| 22 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 23 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 24 | 10 | 100% | 163.7333 | 0% | 91% | 56% | 164% |
| 25 | 10 | 100% | 162.9833 | 2% | 91% | 56% | 162% |
| 26 | 10 | 100% | 161.4476 | 2% | 91% | 56% | 162% |
| 27 | 10 | 100% | 161.4476 | 0% | 91% | 56% | 162% |
| 28 | 9 | 100% | 165.8286 | 4% | 91% | 57% | 160% |
| 29 | 10 | 100% | 163.7333 | 2% | 91% | 56% | 164% |
| 30 | 9 | 100% | 166.7333 | 2% | 91% | 57% | 160% |
| 31 | 9 | 100% | 168.8444 | 2% | 91% | 56% | 161% |
| 32 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 33 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 34 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 35 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 36 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 37 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 38 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 39 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |

| Number of Eigenvectors | Number of Clusters | Accuracy | SSE | % change in cluster membership | within-Cluster similarity | Inter-cluster similarity | Ratio of intra/ inter similarity |
|---|---|---|---|---|---|---|---|
| 40 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 41 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 42 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 43 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 44 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 45 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 46 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 47 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 48 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 49 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 50 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 51 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 52 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 53 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 54 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 55 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 56 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 57 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |
| 58 | 9 | 100% | 168.8444 | 0% | 91% | 56% | 161% |

**Table 4-9 (continued from previous page)**

**Table 4-10 - Accuracy Results for the Congressional Votes data set using SCCADDS2 and different values of k (number of eigenvectors ) – threshold for the merge algorithm is 0.5.**

| Number of Eigenvectors | Number of Clusters | Accuracy | SSE | % change in cluster membership | within-Cluster similarity | Inter-cluster similarity | Ratio of intra/ inter similarity |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 88% | 2387.433 | 0% | 81% | 44% | 185% |
| 2 | 4 | 88% | 2294.5 | 4% | 82% | 71% | 116% |
| 3 | 5 | 88% | 2258.202 | 2% | 82% | 71% | 115% |
| 4 | 7 | 89% | 2156.016 | 4% | 83% | 68% | 122% |
| 5 | 8 | 89% | 2106.053 | 4% | 83% | 70% | 119% |
| 6 | 11 | 88% | 2097.193 | 10% | 83% | 68% | 123% |
| 7 | 15 | 89% | 2054.467 | 5% | 84% | 65% | 130% |
| 8 | 17 | 89% | 2038.193 | 4% | 84% | 63% | 134% |
| 9 | 18 | 90% | 2029.583 | 4% | 84% | 61% | 138% |
| 10 | 14 | 89% | 2048.602 | 2% | 84% | 64% | 131% |
| 11 | 17 | 89% | 2025.207 | 2% | 84% | 66% | 127% |
| 12 | 18 | 89% | 2025.819 | 4% | 84% | 66% | 128% |
| 13 | 18 | 89% | 2024.659 | 1% | 84% | 65% | 129% |
| 14 | 20 | 89% | 1982.002 | 1% | 84% | 62% | 135% |
| 15 | 20 | 89% | 1998.503 | 2% | 84% | 65% | 129% |
| 16 | 21 | 90% | 1990.604 | 3% | 84% | 63% | 133% |
| 17 | 19 | 89% | 1998.892 | 3% | 84% | 66% | 128% |
| 18 | 21 | 90% | 1989.141 | 3% | 84% | 64% | 131% |
| 19 | 24 | 90% | 1950.262 | 3% | 85% | 64% | 133% |
| 20 | 23 | 88% | 1966.786 | 3% | 85% | 66% | 129% |
| 21 | 23 | 89% | 1944.15 | 3% | 85% | 65% | 130% |
| 22 | 25 | 89% | 1933.963 | 3% | 85% | 64% | 132% |
| 23 | 25 | 89% | 1928.332 | 3% | 85% | 64% | 132% |
| 24 | 25 | 89% | 1936.26 | 2% | 85% | 64% | 132% |
| 25 | 24 | 89% | 1931.339 | 5% | 85% | 66% | 129% |
| 26 | 27 | 90% | 1916.084 | 3% | 85% | 63% | 134% |
| 27 | 27 | 90% | 1925.526 | 2% | 85% | 64% | 133% |
| 28 | 26 | 89% | 1926.096 | 3% | 85% | 64% | 133% |
| 29 | 27 | 90% | 1920.838 | 3% | 85% | 64% | 132% |
| 30 | 27 | 90% | 1930.87 | 2% | 85% | 64% | 132% |
| 31 | 23 | 89% | 1975.411 | 1% | 84% | 66% | 129% |
| 32 | 24 | 89% | 1976.322 | 1% | 84% | 65% | 130% |
| 33 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 34 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 35 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 36 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 37 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 38 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 39 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |

| Number of Eigenvectors | Number of Clusters | Accuracy | SSE | % change in cluster membership | within-Cluster similarity | Inter-cluster similarity | Ratio of intra/ inter similarity |
|---|---|---|---|---|---|---|---|
| 40 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 41 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 42 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 43 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 44 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 45 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 46 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 47 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |
| 48 | 24 | 89% | 1976.322 | 0% | 84% | 65% | 130% |

Table 4-10 **continued from previous page**

**Table 4-11 -Results for the Mushroom data set using SCCADDS2 and different values of k (number of eigenvectors ) – threshold for the merge algorithm is 0.5.**

| Number of Eigenvectors | Number of Clusters | Accuracy | SSE | % change in cluster membership | within-Cluster similarity | Inter-cluster similarity | Ratio of intra/ inter similarity |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 89% | 78709.51 | 0% | 75% | 71% | 105% |
| 2 | 5 | 89% | 63591.46 | 1% | 80% | 61% | 131% |
| 3 | 8 | 89% | 58878.53 | 1% | 82% | 61% | 134% |
| 4 | 10 | 90% | 60405.67 | 14% | 81% | 59% | 137% |
| 5 | 10 | 91% | 52742.82 | 28% | 84% | 59% | 142% |
| 6 | 12 | 91% | 50491.15 | 6% | 85% | 59% | 143% |
| 7 | 13 | 90% | 48086.78 | 9% | 85% | 57% | 150% |
| 8 | 13 | 90% | 47903.93 | 1% | 85% | 57% | 149% |
| 9 | 14 | 91% | 45298.17 | 4% | 86% | 57% | 151% |
| 10 | 15 | 94% | 43907.57 | 5% | 87% | 57% | 152% |
| 11 | 16 | 94% | 43145.09 | 1% | 87% | 58% | 151% |
| 12 | 16 | 95% | 42265.43 | 5% | 87% | 58% | 151% |
| 13 | 17 | 97% | 40445.7 | 4% | 88% | 56% | 156% |
| 14 | 18 | 97% | 38416.82 | 3% | 89% | 55% | 161% |
| 15 | 18 | 98% | 38303.36 | 2% | 89% | 55% | 161% |
| 16 | 19 | 98% | 37597.83 | 1% | 89% | 55% | 161% |
| 17 | 19 | 99% | 37438.57 | 0% | 89% | 55% | 161% |
| 18 | 20 | 99% | 37942.48 | 1% | 89% | 54% | 164% |
| 19 | 21 | 99% | 37401.47 | 2% | 89% | 56% | 159% |
| 20 | 22 | 99% | 37278.51 | 0% | 89% | 56% | 160% |
| 21 | 23 | 99% | 37475.21 | 1% | 89% | 56% | 158% |
| 22 | 23 | 99% | 37477.69 | 0% | 89% | 56% | 158% |
| 23 | 23 | 99% | 37475.21 | 0% | 89% | 56% | 158% |
| 24 | 23 | 99% | 37477.69 | 0% | 89% | 56% | 158% |
| 25 | 23 | 99% | 37477.69 | 0% | 89% | 56% | 158% |
| 26 | 24 | 99% | 37459.21 | 0% | 89% | 56% | 158% |
| 27 | 24 | 99% | 37459.21 | 0% | 89% | 56% | 158% |
| 28 | 25 | 99% | 37454.46 | 0% | 89% | 56% | 157% |
| 29 | 25 | 99% | 37451.41 | 0% | 89% | 56% | 157% |
| 30 | 25 | 99% | 37249.05 | 2% | 89% | 56% | 157% |
| 31 | 25 | 99% | 37451.62 | 0% | 89% | 56% | 157% |
| 32 | 25 | 99% | 37451.62 | 0% | 89% | 56% | 157% |
| 33 | 25 | 99% | 37084.04 | 5% | 89% | 56% | 159% |
| 34 | 26 | 99% | 37019.4 | 1% | 89% | 56% | 158% |
| 35 | 26 | 99% | 37124.22 | 2% | 89% | 56% | 158% |
| 36 | 27 | 99% | 37158.28 | 1% | 89% | 57% | 157% |
| 37 | 26 | 99% | 37108.93 | 0% | 89% | 56% | 158% |
| 38 | 26 | 99% | 37200.77 | 0% | 89% | 57% | 157% |
| 39 | 27 | 99% | 37021.21 | 1% | 89% | 56% | 158% |

| Number of Eigenvectors | Number of Clusters | Accuracy | SSE | % change in cluster membership | within-Cluster similarity | Inter-cluster similarity | Ratio of intra/ inter similarity |
|---|---|---|---|---|---|---|---|
| 40 | 28 | 99% | 36970.98 | 1% | 89% | 56% | 158% |
| 41 | 28 | 99% | 37132.95 | 1% | 89% | 55% | 161% |
| 42 | 28 | 99% | 37091.97 | 0% | 89% | 55% | 162% |
| 43 | 27 | 99% | 37196.05 | 0% | 89% | 56% | 158% |
| 44 | 28 | 99% | 37114.51 | 0% | 89% | 55% | 161% |
| 45 | 30 | 99% | 37104.27 | 0% | 89% | 56% | 160% |
| 46 | 29 | 99% | 37064.2 | 0% | 89% | 55% | 161% |
| 47 | 29 | 99% | 37060.34 | 0% | 89% | 55% | 162% |
| 48 | 30 | 99% | 36443.74 | 0% | 89% | 55% | 161% |

**Table 4-11 continued from previous page**

**Table 4-12 - SCCADDS1 - Accuracy Results on Synthetic Data Sets – Merge Threshold is .0.5.**

| Data Set Name | Number of Eigen-vectors | Number of Clusters | Accuracy | % of data objects changing clusters |
|---|---|---|---|---|
| DS1 | 2 | 2 | 39% | 0% |
| | 3 | 5 | 60% | 3% |
| | 4 | 5 | 79% | 20% |
| | 5 | 5 | 100% | 21% |
| | 6 | 5 | 100% | 0% |
| | 7 | 5 | 100% | 0% |
| | 8 | 5 | 100% | 0% |
| | 9 | 5 | 100% | 0% |
| | 10 | 5 | 100% | 0% |
| DS2 | 2 | 2 | 21% | 0% |
| | 3 | 10 | 51% | 4% |
| | 4 | 10 | 68% | 31% |
| | 5 | 10 | 75% | 12% |
| | 6 | 10 | 71% | 4% |
| | 7 | 10 | 73% | 6% |
| | 8 | 10 | 89% | 27% |
| | 9 | 10 | 100% | 11% |
| | 10 | 10 | 100% | 0% |
| DS3 | 2 | 2 | 21% | 0% |
| | 3 | 10 | 51% | 1% |
| | 4 | 10 | 59% | 26% |
| | 5 | 10 | 57% | 12% |
| | 6 | 10 | 84% | 31% |
| | 7 | 10 | 91% | 7% |
| | 8 | 10 | 91% | 0% |
| | 9 | 10 | 91% | 0% |
| | 10 | 10 | 100% | 9% |
| DS4 | 2 | 2 | 21% | 0% |
| | 3 | 10 | 48% | 5% |
| | 4 | 10 | 58% | 28% |
| | 5 | 10 | 59% | 15% |
| | 6 | 10 | 67% | 22% |
| | 7 | 10 | 76% | 13% |
| | 8 | 10 | 98% | 21% |
| | 9 | 10 | 90% | 2% |
| | 10 | 10 | 100% | 10% |
| DS5 | 2 | 2 | 21% | 0% |
| | 3 | 10 | 37% | 10% |
| | 4 | 10 | 46% | 12% |
| | 5 | 10 | 61% | 20% |
| | 6 | 10 | 71% | 12% |
| | 7 | 10 | 90% | 19% |
| | 8 | 10 | 91% | 10% |
| | 9 | 10 | 100% | 10% |
| | 10 | 10 | 100% | 0% |
| | 11 | 10 | 100% | 0% |
| | 12 | 10 | 100% | 0% |
| | 13 | 10 | 100% | 0% |
| | 14 | 10 | 100% | 0% |
| | 15 | 10 | 100% | 0% |
| | 16 | 10 | 100% | 0% |
| | 17 | 10 | 100% | 0% |
| | 18 | 10 | 100% | 0% |
| | 19 | 10 | 100% | 0% |
| | 20 | 10 | 100% | 0% |
| | 11 | 10 | 100% | 0% |
| | 12 | 10 | 100% | 0% |
| | 13 | 10 | 100% | 0% |
| | 14 | 10 | 100% | 0% |
| | 15 | 10 | 100% | 0% |
| | 16 | 10 | 100% | 0% |
| | 17 | 10 | 100% | 0% |
| | 18 | 10 | 100% | 0% |
| | 19 | 10 | 100% | 0% |
| | 20 | 10 | 100% | 0% |

**Table 4-13 - SCCADDS2 - Accuracy Results on Synthetic Data Sets – Merge threshold is 0.5.**

| Data Set Name | Number of Eigen-vectors | Number of Clusters | Accuracy | % of data objects changing clusters |
|---|---|---|---|---|
| DS1 | 1 | 2 | 38% | 0% |
| | 2 | 5 | 61% | 5% |
| | 3 | 5 | 79% | 19% |
| | 4 | 5 | 100% | 20% |
| | 5 | 5 | 100% | 0% |
| | 6 | 5 | 100% | 0% |
| | 7 | 5 | 100% | 0% |
| | 8 | 5 | 100% | 0% |
| | 9 | 5 | 100% | 0% |
| | 10 | 6 | 100% | 0% |
| DS2 | 1 | 2 | 21% | 0% |
| | 2 | 8 | 50% | 5% |
| | 3 | 10 | 64% | 11% |
| | 4 | 10 | 66% | 7% |
| | 5 | 10 | 68% | 8% |
| | 6 | 10 | 70% | 7% |
| | 7 | 10 | 82% | 12% |
| | 8 | 10 | 90% | 9% |
| | 9 | 10 | 99% | 11% |
| | 10 | 10 | 99% | 0% |
| DS3 | 11 | 10 | 99% | 0% |
| | 12 | 10 | 99% | 0% |
| | 13 | 10 | 99% | 0% |
| | 14 | 10 | 99% | 0% |
| | 15 | 10 | 99% | 0% |
| | 16 | 10 | 99% | 0% |
| | 17 | 10 | 99% | 0% |
| | 18 | 10 | 99% | 0% |
| | 19 | 10 | 99% | 0% |
| | 20 | 10 | 99% | 0% |
| DS4 | 1 | 2 | 21% | 0% |
| | 2 | 8 | 45% | 4% |
| | 3 | 10 | 56% | 15% |
| | 4 | 10 | 52% | 8% |
| | 5 | 10 | 64% | 16% |
| | 6 | 10 | 74% | 18% |
| | 7 | 10 | 87% | 16% |
| | 8 | 10 | 91% | 4% |
| | 9 | 10 | 100% | 9% |
| | 10 | 10 | 100% | 0% |
| DS5 | 11 | 10 | 100% | 0% |
| | 12 | 10 | 100% | 0% |
| | 13 | 10 | 100% | 0% |
| | 14 | 10 | 100% | 0% |
| | 15 | 10 | 100% | 0% |
| | 16 | 10 | 100% | 0% |
| | 17 | 10 | 100% | 0% |
| | 18 | 10 | 100% | 0% |
| | 19 | 10 | 100% | 0% |
| | 20 | 10 | 100% | 0% |
| DS6 | 1 | 2 | 21% | 0% |
| | 2 | 7 | 40% | 11% |
| | 3 | 10 | 46% | 10% |
| | 4 | 10 | 53% | 13% |
| | 5 | 10 | 63% | 11% |
| | 6 | 10 | 71% | 10% |
| | 7 | 10 | 99% | 29% |
| | 8 | 10 | 99% | 2% |
| | 9 | 10 | 100% | 1% |
| | 10 | 10 | 100% | 0% |
| DS7 | 11 | 10 | 100% | 0% |
| | 12 | 10 | 100% | 0% |
| | 13 | 10 | 100% | 0% |
| | 14 | 10 | 100% | 0% |
| | 15 | 10 | 100% | 0% |
| | 16 | 10 | 100% | 0% |
| | 17 | 10 | 100% | 0% |
| | 18 | 10 | 100% | 0% |
| | 19 | 10 | 100% | 0% |
| | 20 | 10 | 100% | 0% |

**Table 4-14 – The accuracy results, inter-cluster similarity and intra-similarity for the synthetic data sets DS5. The threshold used for the merge algorithm is 0.5. DS5 has a noise ratio of 10%**

| Number of Eigenvectors | Number of Clusters | Accuracy | SSE | % change in cluster membership | within-Cluster similarity | Inter-cluster similarity | Ratio of intra/ inter similarity |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 21% | 85282.79 | 0% | 38% | 41% | 92% |
| 2 | 7 | 40% | 78036.16 | 11% | 47% | 36% | 131% |
| 3 | 10 | 46% | 74735.61 | 10% | 50% | 29% | 169% |
| 4 | 10 | 53% | 71591.04 | 13% | 53% | 25% | 214% |
| 5 | 10 | 63% | 67901.76 | 11% | 56% | 18% | 310% |
| 6 | 10 | 71% | 64948.7 | 10% | 59% | 16% | 375% |
| 7 | 10 | 99% | 55957.14 | 29% | 66% | 15% | 439% |
| 8 | 10 | 99% | 56093.9 | 2% | 66% | 15% | 433% |
| 9 | 10 | 100% | 55417.07 | 1% | 67% | 15% | 451% |
| 10 | 10 | 100% | 55417.07 | 0% | 67% | 15% | 451% |
| 11 | 10 | 100% | 55417.07 | 0% | 67% | 15% | 451% |
| 12 | 10 | 100% | 55417.07 | 0% | 67% | 15% | 451% |
| 13 | 10 | 100% | 55417.07 | 0% | 67% | 15% | 451% |
| 14 | 10 | 100% | 55417.07 | 0% | 67% | 15% | 451% |
| 15 | 10 | 100% | 55417.07 | 0% | 67% | 15% | 451% |
| 16 | 10 | 100% | 55417.07 | 0% | 67% | 15% | 451% |
| 17 | 10 | 100% | 55417.07 | 0% | 67% | 15% | 451% |
| 18 | 10 | 100% | 55417.07 | 0% | 67% | 15% | 451% |
| 19 | 10 | 100% | 55417.07 | 0% | 67% | 15% | 451% |
| 20 | 10 | 100% | 55417.07 | 0% | 67% | 15% | 451% |

# Chapter 5 - Clustering Using NIBRS

In this chapter, we apply the algorithms presented in this thesis to a specific application domain to examine if the algorithms will detect clusters in the data. We believe our algorithms are particularly well suited to criminal justice data applications where most of the attributes are categorical and the number of attributes is often small relative to the number of data objects. Data mining and clustering of criminal justice data is of increasing importance to both the criminal justice research community and law enforcement practitioners (Chen et al., 2004). For our evaluation, we use the FBI's National Incident Based Reporting System (NIBRS). NIBRS contains detailed information regarding crime incidents and is one of the few crime data repositories that has the detailed information researchers and law enforcement practitioners need to gain an in-depth understanding of crimes and criminal behavior. NIBRS contains data for over 29 million criminal incidents.

## 1 Introduction

Our goal is to apply our algorithms to a crime-based data collection in order to evaluate the effectiveness of the algorithms in discovering clusters that occur in actual crime data and develop an effective tool that allows criminal justice researchers to gain insight into the nature and causes of common criminal activity. Data collected through NIBRS contain a collection of attributes related to criminal incidents occurring in the United States. Almost all of these attributes are categorical which makes the data suitable for our algorithms. The data is collected from local, state, and federal automated systems

throughout the nation based on recommended procedures and guidelines provided by the United States' Federal Bureau of Investigation (FBI).

We tested CATS, SCCADDS1, and SCCADDS2 on a selected subset of data from NIBRS. Our objective is to profile young victims (10 years or younger) of criminal offenses. We wanted to determine if the algorithms could find groups of victims that share some common characteristics. The algorithms successfully discovered offense-based clusters where the victims share some characteristics that mostly occur with certain offenses. For example, offenses that typically occurred against females were clustered together. In addition to offense-based clusters, the algorithms discovered clusters where the victims had certain characteristics in common regardless of the crime committed. For example, the algorithms grouped victims that are of the same race in one cluster and victims with certain types of injuries in another cluster.

The organization of the remainder of this chapter is the following. In Section 2, we provide an overview of NIBRS and the data it contains. We discuss the data segments used to organize NIBRS data and discuss how segments are linked. In Section 3, we describe our tests and the results. Section 4 contains concluding remarks.

# *2 NIBRS*

## 2.1 Overview

NIBRS is part of Uniform Crime Reporting Program (UCR). The program is managed by the Federal Bureau of Investigation (FBI) of the U.S. Department of Justice. UCR is a national crime-based data collection program that was initiated in January 1930. In the late 1970's, the law enforcement community realized the need for a more comprehensive reporting system to meet crime challenges of the 21st century. As such, a multi-year

study was initiated to evaluate the data collected at that time, and the needs of the law enforcement community and other communities with interest in crime data. The specification and guidelines for NIBRS were developed based on the Abt Associates Inc. report entitled "Blueprint for the Future of the Uniform Crime Reporting Program" issued in May 1985 (UCR, NIBRS, Volume 1, 2000). The first pilot for NIBRS was implemented by the South Carolina Law Enforcement Division (SLED) in 1987. More guidelines and specification were incorporated into NIBRS based on the pilot study by SLED and further recommendations from the law enforcement community.

The goals of NIBRS were to improve crime reporting in terms of quantity, quality and timeliness and to improve the methodology of "compiling, analyzing, auditing, and publishing" the collected data (UCR, NIBRS, Volume 1, 2000). The main difference between NIBRS and the traditional UCR system is that NIBRS is an incident-based reporting system while UCR collects summary data regarding offenses. Traditional UCR reporting (non-NIBRS) contains crime-based summarized reports (aggregate statistics) on 8 major offenses (criminal homicide, forcible rape, robbery, aggravated assault, burglary, larceny-theft, motor vehicle theft, arson). The UCR program also provides aggregate statistics on persons arrested, hate crimes, and law enforcements agents killed and assaulted (Uniform Crime Reporting Handbook, 2004). On the other hand, NIBRS data contain the details of each crime incident reported to the FBI, e.g., information on the arrestee, offender, property and victim associated with the incident. NIBRS crime data consists of 22 offense categories that are made up of 46 specific crimes for Group A and 11 offenses for Group B. Offenses in NIBRS are divided into two groups: Group A and Group B. Offenses in Group A are typically more serious than offenses in Group B. For

each incident involving a Group A offense, collected data is organized into six segments: the Administrative, Offense, Property, Victim, Offender, and Arrestee segments.

Since NIBRS is not a mandatory crime reporting system, not all law enforcement agencies contribute data to NIBRS. In 2004, 5,271 law enforcement agencies participated in NIBRS[6] and about 17,000 agencies contributed non-NIBRS crime-based summarized reports (Uniform Crime Reporting Handbook, 2004). In 2004, the data collected for NIBRS represented only 16% of data collected by the UCR program[6]. (However, more law enforcement agencies are in the process of collecting data for NIBRS.) For additional information regarding NIBRS and its guidelines, please see (UCR, NIBRS, Volume 1, 2000; UCR, NIBRS, Volume 2, 1992; UCR, NIBRS, Volume 3, 1997; UCR, NIBRS, Volume 4, 1999). Many states now have NIBRS compliant crime data systems that produce monthly NIBRS reports that are uploaded to the FBI.

## 2.2 NIBRS Data Segments

In this section, we provide an overview of the data stored in NIBRS. Detailed discussions can be found in these references (UCR, NIBRS, Volume 1, 2000; Akiyama, 1998; Akiyama and Nolan, 1999; Snyder, 1999; Finklehor and Ormrod, 2004).

All data reporting in NIBRS is per a crime incident. In NIBRS, an incident is defined as "one or more offenses committed by the same offender, or group of offenders acting in concert, at the same time and place"( UCR, NIBRS, Volume 1, 2000). As mentioned previously, offenses in NIBRS are classified into two groups: Group A and Group B. Group A offenses are typically more serious and significant than Group B offenses, and as such, more reporting is required for Group A offenses. Only an arrest report is

[6] Data obtained from FBI web site at http://www.fbi.gov/ucr/ucr htm

required for a Group B offense. Examples of Group B offenses are runaway, trespassing of real property, and non-violent family offenses. NIBRS uses the term "segment" to refer to a collection of data elements that describe an entity or an event. For Group A offenses, NIBRS collects data in six segments and they are:

- **Administrative Segment** contains control information regarding an incident such as the number of offenses and the number of victims.
- **Offense Segment** contains information regarding the offense such as an offense code, location of the offense, and weapon used in the offense. There are one or more offense segments per incident.
- **Property Segment** contains information regarding the property affected by the incident. There can be one or more property segments associated with a crime incident.
- **Victim Segment** contains information regarding the victims. There can be one or more victim segments associate with a crime incident.
- **Offender Segment** contains information regarding the offenders. There can be one or more offender segments associated with a crime incident.
- **Arrestee Segment** contains information regarding the arrestees. There can be one or more arrestee segments associated with a crime incident.

Only an arrestee segment is needed for a Group B offense. All the segments related to a crime incident are linked together using the Originating Agency Identifier (ORI) (i.e. law enforcement agency reporting the incident) and the incident number. Also, there are fields in some segments that identify other segments. For example, the Victim Segment contains fields that relate a victim to offenders and as such the offender ID (a

sequence number that uniquely identify each offender within an incident) are included in the Victim Segment. A victim can be related to multiple offenders; there are 20 fields in the Victim Segment that can contain the relationship between the victim and up to 10 offenders.

Other segments used in NIBRS are Window Segments and Batch Segments. Window Segments are used to record incomplete Group A incident reports or incidents that occurred prior to when the agency began reporting to NIBRS. Batch Segments contain information regarding law enforcements agencies such as ORI numbers and locations.

NIBRS data can be obtained from the FBI in one variable-length file with multiple record layouts where each layout corresponds to one of the segments of NIBRS. However, the National Archive of Criminal Justice Data (NACJD)[7] extracts the NIBRS data segments from the flat file and stores them into separate data files. These files are available at NACJD website for downloading.

At John Jay College of Criminal Justice, we have created a relational database using Oracle that houses all data segments as well as all code tables used in NIBRS. The database contains NIBRS data from 1994 to 2005. Figure 5-1 presents an entity-relationship model of the data segments used for group A offenses. Note that we excluded the code tables from Figure 5-1 to make it easier to see key relationships among the entities. Also, we only show the primary relationship between the tables. For example, the Victim segment contains information about multiple offenders, multiple offenses and multiple injury types. The relationships between the Victim Segment and the Offender and Offenses Segments are not shown in Figure 5-1.

[7] NACJD website for NIBRS is http://www.icpsr.umich.edu/NACJD/NIBRS/.

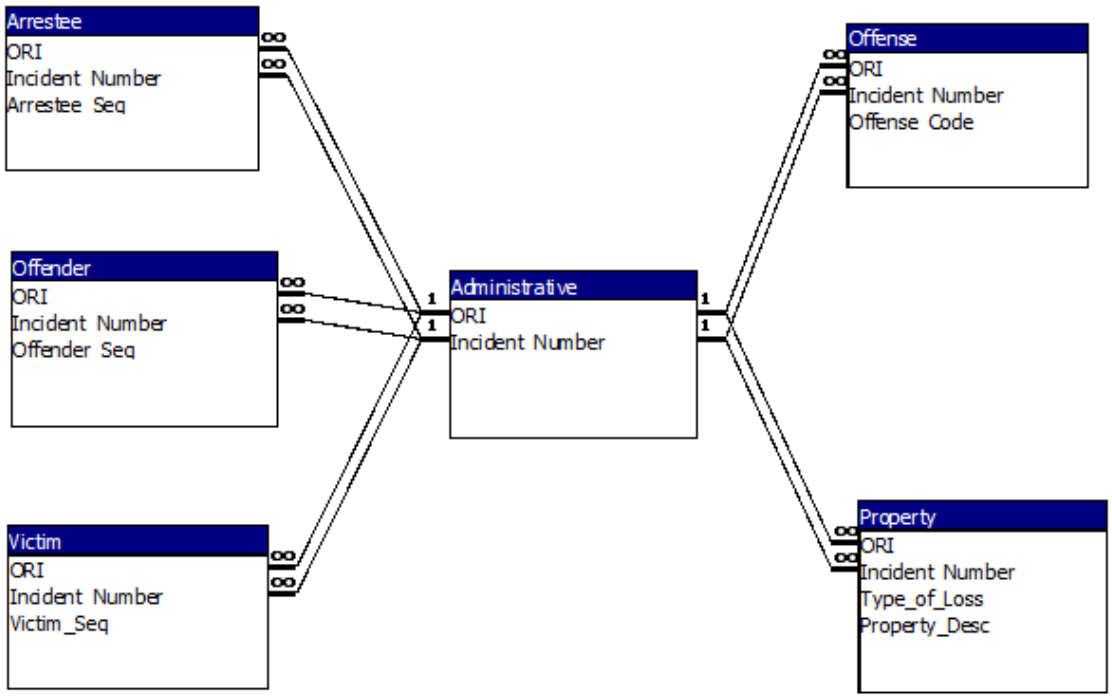


**Figure 5-1 - NIBRS relationships for Group A offenses**

## 3 Clustering NIBRS using CATS and SCCADDS

The NIBRS database contains millions of records for each data segment. For example, there are over 29 million incidents for years 1994 through 2005 in NIBRS. The Victim Segment contains 5,067,759 victim records for just 2005. For our evaluation, we selected the Victim Segment to perform clustering for victims that are 10 years of age or younger. We extracted data for 4 states (Arizona, Arkansas, Colorado, and Connecticut) that represent a cross section of the United States. The data are from 2005, the most recent year for which NIBRS data was available. Our test data set contains 4,409 victim records.

## 3.1 Data Description

The Victim Segment contains various data about the victims involved in a crime incident. There are three repeating groups of data in the Victim Segment. The three groups are for the offense code, injury type, and victim relationship to the offenders. A victim can be related to one or more offenses, have zero or more injuries, and be related to zero or more offenders. Figure 5-2 shows the relationships between the Victim Segments and these repeating groups. Figure 5-2 shows a relational data model in third-normal (Harrington, 2002; Sumathi and Esakkirajan, 2007) form for the Victim Segment and the related segments (Offense and Offender). The figure shows the following relationships:

- Table "Victim_Offenses" is a link table to implement a 1-to-many relationship between the Victim Segment and the Offense Segment. This table ensures the integrity of the relationship. In other words, a victim can only be related to an offense that was committed as part of the criminal incident that the victim is part of. A victim and an offense can only be related if they are part of the same criminal incident. Since an offense can affect multiple victims, this table also implements a 1-to-many relationship from an offense in a criminal incident to the victims of the offense. In short, the table is a many-many relationship between the Victim Segment and the Offense Segment.
- Table "Victim_to_Offender_Rel" is a link table that implements many-to-many relationship between the Victims and Offenders. The table would contain information on the type of relationship between a victim and an offender (i.e. a child, step-child, etc.).

- Table “Victim_Injury” implements a 1-to-many relationship between a victim and injury types.

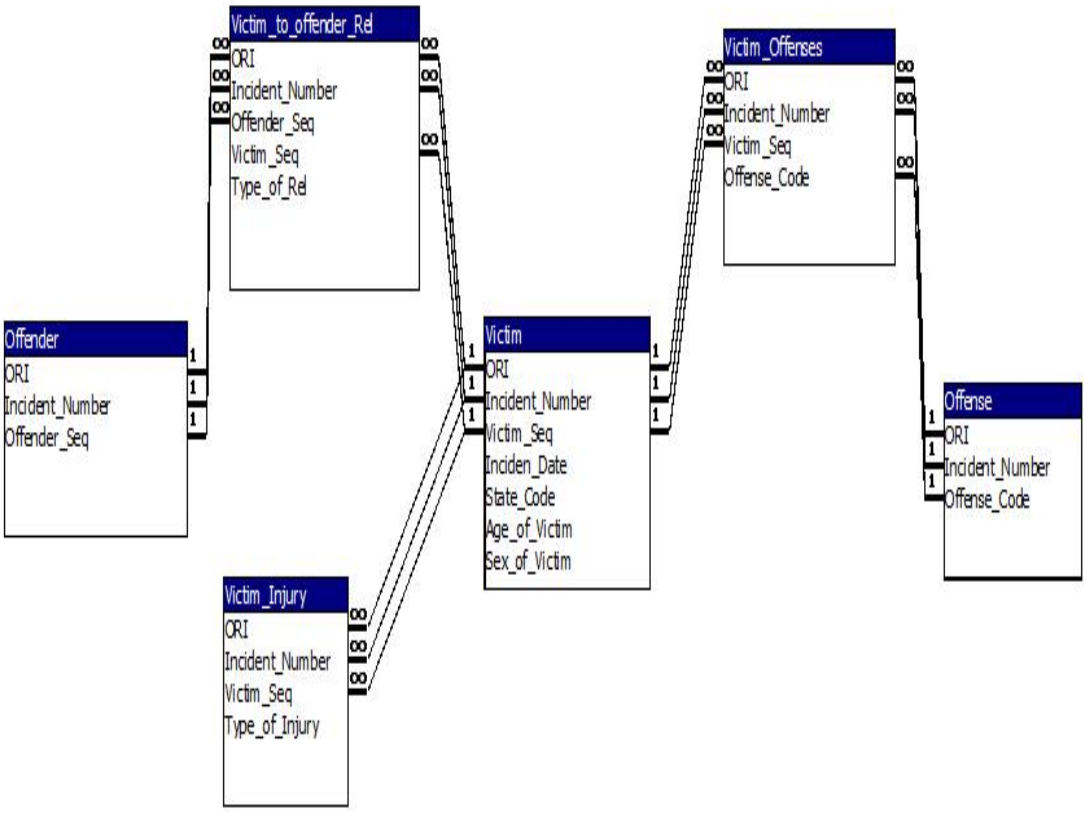


**Figure 5-2 - Relational data model of the Victim, Offense and Offender Segments**

For our evaluation, we excluded attributes that occurred in every record, attributes related to geographical location, attributes that rarely contained a value, and attributes that uniquely identified each record (i.e incident number, date). We used the following attributes from the Victim segment, which typically are of interest to criminal justice researchers.

- Age of victim
- Sex of Victim
- Race of Victim

- Ethnicity of Victim
- Residency Status of Victim
- Offense Codes
- Injury Types
- Relationship to the offender

We tested CATS, SCCADDS1, and SCCADDS2 on the victim data. There are 66 attribute categories (dimensions) in the attribute proximity matrix. Table 5-1 shows the number of attribute categories for each attribute. Note that we created a new field to indicate whether a victim is related to the offender since we were not interested in clustering based on victim-offender relationship.

| **Attribute Name** | **Number of Categories** |
|---|---|
| Age of Victim | 12 |
| Sex of Victim | 3 |
| Race of Victim | 5 |
| Ethnicity of Victim | 3 |
| Residency Status of Victim | 3 |
| Offense Code | 31 |
| Injury Type | 8 |
| Relationship to Offender | 1 |
| **Total** | **66** |

**Table 5-1 – Number of attribute categories in each attribute for the Victim Segment**

## 3.2 Evaluation Methodology

For the test data set extracted from NIBRS, we do not have any pre-defined clusters that we can use to evaluate the results of our algorithms. Evaluating clustering quality and the usefulness of the clusters in the absence of pre-defined clusters can be subjective as it may depend on the perspective of the evaluator. For example, one evaluator may be interested in clustering offenses by the type of injury the victims sustained while another

may be interested in clustering the offenses by victims' age. Our evaluation will be quantitative using the occurrence frequency of the attributes categories in the test data set. We will not evaluate if the algorithms found *every possible cluster* in the data set. However, we will evaluate if the clusters are valid; that is if the victim records grouped together do indeed share common characteristics that differentiate them from the rest of the population in the test data set. We will use Recall and Precision measures to validate the clusters. Assuming knowledge of the pre-defined clusters, Recall measures the percentage of data objects that are clustered properly (recalled into clusters). On the other hand, Precision measures the percentage of the data objects in a cluster that actually belongs to a pre-defined cluster. High Recall percentages indicate that a clustering algorithm discovered most of the clusters in a data set while high Precision percentages indicate that the clusters are very specialized, i.e. clusters mostly contain data objects belonging to one pre-defined cluster. In the remainder of this chapter we will use the terms "data objects" and "records" interchangeably. Our evaluation methodology is as follows:

- Create a profile for the test data set that contains the occurrence frequency of each attribute category in the test data set. This profile is presented in Table 5-2. We will use this profile to determine possible clusters in the test data set and to verify the significance of the clusters found by the algorithms.
- For every cluster discovered by the algorithms, we will create a profile similar to the profile shown in Table 5-2. These profiles will show which attribute categories dominate the cluster and as such can be used to describe a cluster. We will use these attribute categories to create a *description* of the cluster.

- Once we obtain a *description* of the clusters, we can then find the victim records in the test data set that have these attribute categories. We will consider this group of records that we find to be a cluster that exists in the test data set and as such it will be treated similar to a pre-defined cluster for the purpose of our evaluation.
- We will use Recall and Precision measures to validate the clusters.

| Offense Code | Occ. Freq | Age of Victim | Occ. Freq | Sex of Victim | Occ. Freq | Race of Victim | Occ. Freq | Ethinicity of Victim | Occ. Freq | Resident Status of Victim | Occ. Freq | Injury Type | Occ. Freq | Related | Occ. Freq |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 11D | 926 | BB | 132 | M | 2,140 | U | 342 | U | 1,511 | U | 325 | N | 2,248 | R | 3,684 |
| 13B | 1,263 | 1 | 232 | F | 2,225 | W | 3,234 | N | 2,334 | R | 3,509 | M | 1,026 | | |
| 100 | 143 | 4 | 392 | U | 44 | B | 788 | H | 508 | N | 521 | L | 7 | | |
| 13A | 520 | 8 | 449 | | | A | 29 | | | | | I | 42 | | |
| 23H | 255 | 9 | 597 | | | I | 16 | | | | | O | 47 | | |
| 240 | 10 | 10 | 699 | | | | | | | | | B | 30 | | |
| 290 | 123 | 5 | 403 | | | | | | | | | T | 2 | | |
| 26A | 4 | 3 | 337 | | | | | | | | | U | 3 | | |
| 13C | 283 | 2 | 229 | | | | | | | | | | | | |
| 11B | 132 | 7 | 469 | | | | | | | | | | | | |
| 11A | 235 | 6 | 435 | | | | | | | | | | | | |
| 11C | 147 | NB | 15 | | | | | | | | | | | | |
| 36B | 63 | | | | | | | | | | | | | | |
| 220 | 103 | | | | | | | | | | | | | | |
| 26C | 17 | | | | | | | | | | | | | | |
| 23D | 54 | | | | | | | | | | | | | | |
| 120 | 38 | | | | | | | | | | | | | | |
| 200 | 9 | | | | | | | | | | | | | | |
| 23F | 37 | | | | | | | | | | | | | | |
| 23B | 3 | | | | | | | | | | | | | | |
| 09A | 12 | | | | | | | | | | | | | | |
| 250 | 11 | | | | | | | | | | | | | | |
| 23C | 7 | | | | | | | | | | | | | | |
| 36A | 62 | | | | | | | | | | | | | | |
| 09B | 3 | | | | | | | | | | | | | | |
| 23G | 12 | | | | | | | | | | | | | | |
| 23A | 1 | | | | | | | | | | | | | | |
| 26B | 2 | | | | | | | | | | | | | | |
| 210 | 1 | | | | | | | | | | | | | | |
| 280 | 2 | | | | | | | | | | | | | | |
| 270 | 1 | | | | | | | | | | | | | | |

**Table 5-2 – Occurrence frequencies of attribute categories in the test data set selected from the Victim Segment**

Since we will often reference offense codes in our discussion, we provide a list of the offense codes that appeared in our data set and their description in Table 5-3. The offense codes and their description are from NIBRS codebook (ICPSR 4720, 2005).

| Value | Description |
|---|---|
| 200 | Arson |
| | **Assault Offenses** |
| 13A | Aggravated Assault |
| 13B | Simple Assault |
| 13C | Intimidation |
| 510 | Bribery |
| 220 | Burglary/Breaking and Entering |
| 250 | Counterfeiting/Forgery |
| 290 | Destruction/Damage/Vandalism of Property |
| 270 | Embezzlement |
| 210 | Extortion/Blackmail |
| | **Fraud Offenses** |
| 26A | FalsePretenses/Swindle/Confidence Game |
| 26B | Credit Card/Automatic Teller Machine Fraud |
| 26C | Impersonation |
| | **Homicide Offenses** |
| 09A | Murder/Nonnegligent Manslaughter |
| 09B | Negligent Manslaughter |
| 100 | Kidnaping/Abduction |
| | **Larceny/Theft Larceny/Theft Offenses** |
| 23A | Pocket-picking |
| 23B | Purse-snatching |
| 23C | Shoplifting |
| 23D | Theft From Building |
| 23F | Theft From Motor Vechicle Theft of Motor Vehicl |
| 23G | Parts/Accessories |
| 23H | All Other Larceny |
| 240 | Motor Vehicle Theft |
| 120 | Robbery |
| | **Sex Offenses, Forcible** |
| 11A | Forcible Rape |
| 11B | Forcible Sodomy |
| 11C | Sexual Assault With An Object |
| 11D | Forcible Fondling (Indecent Liberties/Child Molest |
| | **Sex Offenses, Nonforcible** |
| 36A | Incest |
| 36B | Statutory Rape |
| 280 | Stolen Property Offenses (Receiving, Selling, Etc.) |

**Table 5-3 - List of Offenses from NIBRS that appeared in our test data set**

## 3.3 CATS

The clusters that were discovered by CATS are based on patterns of attributes that occur frequently in the data set. We executed CATS with a merge threshold of 0.85 and obtained 37 clusters. We chose a high value for the merge threshold because certain attributes had few categories with very high occurrence and co-occurrence frequencies (such as "Sex of the victim", "Race of the victim", and "Relationship to offender") and as such the similarity between the sub-clusters was high (since most of the values for these attributes were the same for most of the victims records).

CATS created a cluster for each set of records that shared a common pattern of attribute categories that distinguished them from other records in the data set. Here are some examples:

- Most of the Victims with minor injuries appeared in one of 5 clusters as follows:
    - Female victims with minor injuries who were subjected to offense 13B and related to the offender. *High Precision.*
    - Male victims with minor injuries who were subjected to offense 13B and related to the offender. *High Precision.*
    - Black non-Hispanic victims with minor injuries and related to the offender. *High Precision.*
    - White non-Hispanic victims with minor injuries who were subjected to offense 13A and related to the offender. *High Precision.*
    - Three-year olds white victims with minor injuries who are related to the offender. *High Precision.*

- Most of the victims of offenses 36A (incest) and 36B (rape) were placed in their respective clusters. These crimes are committed mostly against females. High Recall and *High Precision*
- 9 of 12 victim records that exist in the test data set for offense 09A were placed in one cluster. *High Precision.*
- Three clusters contained only non-resident victims as follows:
    - White male victims with injury type "N" that are non-resident and related to the offender. *High Precision.*
    - White female victims that are non-resident and related to the offender. *High Precision.*
    - Non-Hispanic victims that are non-resident and did not report any injury. *High Precision.*
- Many clusters were based on the Sex of the Victim and a combination of other attributes categories.

It is important to note that each record can only belong to one cluster. This explains why certain victim records are not placed in clusters dedicated to a category of one of their attributes. For example, a victim with minor injuries may not be placed in one of the 5 clusters for victims with minor injuries simply because the victim is also non-resident. This victim record may have other attribute categories that are more similar to the attribute categories in the clusters for non-resident victims than in the clusters for victims with minor injuries.

## 3.4 SCCADDS1 and SCCADDS2

We used the metrics we developed in Chapter 4 to determine the number of eigenvectors to use for our clustering analysis of the Victim data. We looked at the inter-cluster and intra-cluster similarity as well as the percentage of change in cluster membership between different executions of the algorithms using different number of eigenvectors. Overall, we found that the distribution of any single attribute category in the test data set is fairly uniform with respect to all other categories with few exceptions. As a result, the inter-cluster similarity was fairly high. SCCADDS2 performed better than SCCADDS1 in terms of finding more specialized clusters. SCCADDS1 created mainly 2 main clusters that contain the major offense categories (13B and 11D). SCCADDS1 also created some age-related clusters and clustered some offenses into their own clusters. In general SCCADDS2 had better recall and precision than SCCADDS1.

SCCADDS2 was able to isolate groups of victim records that share relatively unique attribute categories when compared to the rest of the records in the test data set. For example, SCCADDS2 grouped 97% of victim records related to offenses 36A (Incest) and 36B (Statutory Rape) in the test data set into one cluster. These offenses typically occur against women and as such 83% of the victims in the cluster were females. It is important to note that SCCADDS1 also discovered this cluster but with lower precision (records for offenses other than 36A and 36B were also placed in the cluster). For SCCADDS2, the precision of this cluster is 100% but for SCCADDS1 is only 84%.

We executed SCCADDS2 multiple times using a different number of eigenvectors in each execution. The percentage of change in cluster membership dropped below 4% when we executed SCCADDS2 using 20 eigenvectors; the percentage of change in

cluster membership remained below 4% for all executions of SCCADDS2 with a number of eigenvectors more than 20 while the number of clusters increased; this indicates that the clusters were splitting (more specialized). Also, the intra-cluster similarity reached 80% and inter-cluster similarity was 65% when SCCADDS2 was executed with 20 eigenvectors. The intra-cluster similarity hovered around 80% for all executions of SCCADDS2 with a larger number of eigenvectors even though the number of clusters increased. On the other hand, the inter-cluster similarity dropped to 57% for 55 clusters. Using 20 eigenvectors, SCCADDS2 produced 37 clusters. Since our clustering is partitional, any victim record had to belong to only one cluster. This resulted in a low recall percentage for some clusters. For example, over 28% of the crime incidents in the test data set are for offense 13B (Simple Assault). Out of the 37 clusters, there is one cluster that is exclusively for offense 13B but the cluster does not contain all victim records associated with offense 13B (Only 43% of the victims' records with offense 13B were grouped into that cluster). The rest of the records were placed in more specialized clusters such as a cluster of 4-year old victims or a cluster for Asian/Pacific Islanders. SCCADDS2 determines the placement of the record based on the characteristics of the individual record and the cluster representatives. We summarize the results of some of the interesting clusters:

- 121 out 125 victims of offenses 36A (Incest) and 36B (Statutory Rape) were grouped in one cluster. The cluster only included offense 36A and 36B. *High recall and high precision.*
- All victims who are Asian/Pacific Islander (Race type A) were grouped into one cluster. The cluster contained 30 victims, 29 of those victims were Asian.

There were 29 victims in the test data set that are Asian/Pacific Islander. *High recall and high precision.*

- 13 out of 16 victims who are American Indians or Alaskan Native (race type I) were placed in one cluster. The cluster contained only American Indians or Alaskan Native victims. *High recall and high precision.*
- 14 out of 15 new-born victims (less than one week old) were grouped in one cluster. They made up 42% of the records in the cluster. *High recall but low precision.*
- 36 out of 42 victim records with internal injury were placed in one cluster. All victim records in the cluster contained injury type "Internal Injury." *High recall and high precision.*
- 222 out 291 victim records in the data set where race and ethnicity are unknown, and the offenses are mostly 11D and 13B were placed in one cluster. They made up 90% of the cluster. *High recall and high precision.*
- One cluster contained white Hispanic victims subjected mostly to offenses 11D, 13A and 13B. *High precision.*
- 5 out of 7 victim records that exist in the test data set for offense 23C (Shoplifting) were placed in their own cluster. *High recall and high precision.*
- 233 out of 255 victim records in the data set for offense 23H (all other Larceny) were grouped in one cluster. Victim records with offense 23H represent 96% of the cluster. *High recall and high precision.*

- 230 out 283 victim records with offense 13C (Intimidation) were placed in one cluster. All victim records in the cluster were for offense 13C. *High recall and high precision.*
- 130 out of 132 victim records with offense 11B were placed in one cluster. The cluster only included 11B offenses. *High recall and high precision.*
- 217 out of 235 victim records with offense 11A were placed in one cluster. 98% of the victim records in the cluster were for offense 11A. *High recall and high precision.*
- 131 out of 147 victim records with offense 11C were placed in one cluster. The cluster only contained records with offense 11C. *High recall and high precision.*
- 49 out of 54 victim records with offense code 23D were placed in one cluster. All victim records in the cluster were for offense 23D. *High recall and high precision.*
- All victim records with offense code 120 were placed in one cluster. *High recall and high precision.*

## 4 Remarks

In this Chapter, we applied our algorithms to criminal justice data. The algorithms successfully extracted groups of data objects that were similar. As discussed in this thesis, categorical data lack a precise definition of distance between data objects and thus there is no natural geometric interpretation for clusters. In turn, clustering algorithms for categorical data use different metrics to determine the similarity between data objects. In our algorithms, as well as in CLICKS and CACTUS, the clusters are created based on the

co-occurrence frequency of attribute categories. Furthermore, a data set may contain several overlapping clusters where each cluster is of interest to data analysts. The NIBRS data set we used contains several overlapping clusters based on our analysis of the occurrence and co-occurrence frequency of attribute categories in the data set. The data objects in the data set can be clustered in several ways and all clusters are equally valid and of interest to researchers. The data set contains clusters where the data objects in each cluster share a combination of common attribute categories that distinguishes the cluster from others in the data set. It also contains clusters of data objects that are characterized by certain attribute categories that rarely occur in the data set. It is important to note that we do not use fuzzy clustering, where a data object may belong to more than one cluster. Consequently, our algorithms cannot find every possible cluster since each data object naturally belongs to more than one cluster but with a different intensity (fuzzy clustering). As such, the algorithms will only show the cluster whose cluster representative is most similar to the data object.

Each algorithm presented here has features that affect the clusterings it finds. CATS, for example, discovered clusters of data objects that contain a combination of attribute categories that occur more often than other attribute categories combinations. CATS uses relative co-occurrence frequencies to ensure that clusters are not based on only the most frequent attribute categories but are based on attribute categories whose co-occurrence and occurrence frequencies are meaningful within the data set. In short, since CATS does not use raw co-occurrence frequency, very frequent categories and very rare categories will have low weights and little impact in the clustering process. On the other hand, SCCADDS algorithms use dimension reduction techniques to filter out noise, and

SCCADDS2, in particular, uses attribute categories covariance. These features enabled the algorithms to find, in addition to clusters similar to those found by CATS, clusters of data objects that contain certain attribute categories not frequently found in other data objects (i.e., clusters based on race ). Some of the clusters found by CATS were also found by SCCADDS but did not have high recall since some of their data objects were clustered into more specialized clusters (i.e., clusters based on a specific offense, or race). Overall, CATS had high precision and mostly high recall for those clusters it found. SCCADDS1 grouped the data mostly into large clusters based on offenses. SCCADDS2 was able to extract those small clusters such as the clusters based on race (Asian/Pacific Islander and American Indians/Alaskan Native) or injury type (Internal Injury). These clusters were based on only one attribute value but were unique enough to warrant their own cluster.

Spectral techniques can be extended to handle fuzzy clustering (See, e.g., Drineas et al., 2004). As part of our future research, we intend to extend SCCADDS to implement fuzzy clustering for categorical data and then apply the new method to cluster criminal justice data. We believe that fuzzy clustering will provide researchers and law enforcement practitioners with a tool that provides a multi-dimensional view of data and illustrates the various ways data objects may clustered.

# Bibliography


1. Abdu, E & Salane, D. (2009). A spectral-based clustering algorithm for categorical data using data summaries, *the proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Workshop on Data Mining using Matrices and Tensors (DMMT'09)..*
2. Akiyama, Y. (1998). Cross-Tabulations and Units of Count with NIBRS Data Elements, Criminal Justice Information Services Division, Federal Bureau of Investigation, Clarksburg, West Virginia, U.S.A.
3. Akiyama, Y. & Nolan, J. (1999). Methods for understanding and analyzing NIBRS data, *Journal of Quantitative Criminology*, 15(2), 225-238.
4. Andritsos, P. (2004). Scalable clustering of categorical data and applications, Ph.D. Dissertation, University of Toronto.
5. Andritsos, P., Tsaparas, P., Miller, R., & Sevcik, K. (2004). LIMBO: scalable clustering of categorical data, *9th International Conference on Extending Database Technology (EDBT).*
6. Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.
7. Balakrishnan, R., & Ranganathan, K. (2000). *A textbook of graph theory.* New York: Springer-Verlag.
8. Barbara, D., Li, Y., & Couto, J. (2002). COOLCAT: an entropy-based algorithm for categorical clustering, *Proceedings of the eleventh international conference on Information and knowledge management*, pp 582-589.

9. Boppana, R. B. (1987). Eigenvalues and graph bisection: An average-case analysis, *Proceedings of the 28th Symposium. Foundations of Computer Science*, pp 280-285.
10. Ben-David, S., Von Luxburg, U., & Pal, D. (2006). A sober look on clustering stability, in G. Lugosi and H. Simon(Eds.), *Proceedings of the 19th Annual Conference on Learning Theory (COLT), pp* 5-19.
11. Berry, M., Dumais, S., & O'brien, G. (1995). Using linear algebra for intelligent information retrieval, *SIAM Review,* 37(4), 573-595.
12. Bobrowski, L. & Bezdek, J.C. (1991). c-Means clustering with the $l_1$ and $l_\infty$ norms, *IEEE Transactions on Systems, Man and Cybernetics*, 21(3), 545–554.
13. Bradford, R.B. (2006). Relationship discovery in large text collections using latent semantic indexing, *SIAM conference on data mining.*
14. Bron C. & Kerbosch J. (1973). Algorithm 457: Finding all cliques of an undirected graph, *Communications of the ACM*, 16(9), 575-577.
15. Chen, H., Chung, W., Xu, J., Wang, G., Qin, Y., & Chau, M. (2004). Crime Data Mining: A General Framework and Some Examples, *IEEE Computer Society*, 37(4), 50-56.
16. Cheng D., Kannan R., Vempala S., & Wang G. (2006). A divide-and-merge methodology for clustering, *ACM Transactions on Database Systems*, 31(4), 1499-1525.
17. Deerwester S., Dumais S., Furnas G., Landauer T., & Harshman R. (1990). Indexing by latent semantic analysis, *Journal of the American Society of Information Science,* 41(6), 391-407.

18. Dhillon, I. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp 269-274.*

19. Ding, C. & He, X (2004). K-means clustering via principal component analysis, *the 21st International Conference on Machine Learning*, Canada.

20. Drineas, P., Frieze, A., Kannan, R., Vempala, S., & Vinay, V. (2004). Clustering large graphs via the singular value decomposition, *Machine Learning,* 56, 9-33.

21. Eckart, C. & Young, G. (1936). The approximation of one matrix by another of lower rank, *Psychometrika*, 1, 183–187.

22. Finklehor, D. and Ormrod, R. (2004). Child pornography: Patterns from NIBRS. Office of Justice Programs, Office of Juvenile Justice and Delinquency Prevention, U.S. Department of Justice, Washington, D.C.

23. Flake, G., Tarjan R., & Tsioutsiouliklis, k. (2004). Graph clustering and minimum cut trees, *Internet Mathematics,* 1(4), 385-408.

24. Frayley, C. & Raftery, A. (1998). How many clusters? Which clustering method? - Answers via model based cluster analysis, Technical Report no. 329, Department of Statistics, University of Washington.

25. Ganti, V., Gehrke, J., & Ramakrishnan, R. (1999). CACTUS clustering categorical data using summaries, *proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and data mining*, pp 73-83.

26. Garey, M. R & Johnson, D. S. (1979). *Computers and Intractability: A guide to the theory of NP-completeness.* New York: W.H. Freeman and Company.

27. Garey, M. J., Johnson, D.S., & Stockmeyer, L. (1976). Some simplified NP-complete graph problems, *Theoretical Computer Science*, 1, 237-267.

28. Gibson, D., Kleinberg, J., & Raghavan, P. (1998). Clustering categorical data: An approach based on dynamical systems, *Proceedings of the 24th International Conference on Very Large Databases.*

29. Golub, G. H. & Van Loan, C. F. (1996). *Matrix computations*, 3rd edition. Baltimore: Johns Hopkins University Press.

30. Guha, S., Rastogi, R., & Shim, K. (1999). ROCK: A robust clustering algorithm for categorical attributes, *proceedings of the 1999 International Conference on Data Engineering (ICDE '99)*, pp 512-521.

31. Hagen, L. & Kahng, A. (1992). New spectral methods for ratio cut partitioning and clustering, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9), 1074-1085.

32. Hand, D. Mannila, H., & Smyth, P. (2001). *Principles of data mining.* Cambridge, MA: The MIT Press.

33. Harrington, J. (2002). *Relational database design, clearly explained, 2nd Edition. San* Francisco, CA: Morgan Kaufmann.

34. Harris, R. (2001). *A primer of multivariate statistics,* 3rd Edition. Mahwah, NJ: Lawrence Erlbaum Associates.

35. Hartigan, J. (1975). *Clustering algorithms.* New York: John Wiley and Sons, Inc.

36. Holzrichter, M. & Oliverira, S. (1999). A graph based method for generating the fiedler vector of irregular problems, *Lecture Notes in Computer Science,* 1586, 978-985.

37. Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components, *Journal of Educational Psychology* 24, 417-441, 488-520.

38. Huang, Z (1998). Extensions to the K-means algorithm for clustering large data sets with categorical values, *Data Mining and Knowledge Discovery,* 2, 283-304.

39. ICPSR 4720. (2005). Inter-University Consortium for Political and Social Research. National Incident–Based Reporting System (NIBRS) Codebook, 2005. United States Department of Justice, Federal Bureau of Investigation.

40. Jain, A.K. & Dubes, R. (1988). *Algorithms for clustering data.* New Jersey: Prentice Hall.

41. Jain, A.K., Murty, M.N., & Flynn, P.J. (1999). Data clustering: A Review, *ACM Computing Surveys,* 31(3), 264-323.

42. Jolliffe, I.T. (2002). *Principal component analysis*, 2nd Edition. New York: Springer.

43. Kannan, R., Vempala, S., & Vetta, A. (2004). On clusterings: good, bad, and spectra, *Journal of the ACM,* 51(3), 497-515.

44. Kantardzic, M. (2003). *Data mining: Concepts, models, methods, and algorithms.* New York: John Wiley and Sons, Inc.

45. Kaufman L. & Rousseeuw P.J. (1990*). Finding groups in data: An introduction to cluster analysis*. New York: John Wiley and Sons, Inc.

46. Khan, S & Kent, S (2007). Computation of initial modes for k-modes clustering algorithm using evidence accumulation, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pp 2784-2789.

47. Kontostathis, A & Pottenger, W. (2002). Detecting patterns in the LSI term-term matrix, Technical Report LU-CSE-02-010, Department of Computer Science and Engineering, Lehigh University.
48. Kontostathis, A & Pottenger, W. (2006). A framework for understanding latent semantic indexing (LSI) performance, *Information Processing and Management,* 42(1), 56-73.
49. Lumley, J.L. (1967). The structure of inhomogeneous turbulent flows, atmospheric turbulence and radio propagation, edited by A.M. Yaglom and V.I. Tatarski, Moscow, pp 166-178.
50. Lutkepohl, H. (1997). *Handbook of matrices.* New York: John Wiley and Sons, Inc.
51. MacQueen, J.B. (1967). Some methods for classification and analysis of multivariate observations, *Proceedings of fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp 281-297.
52. Maimon, O. & Rokach, L. (2005). *Data mining and knowledge discovery handbook*. New York: Springer.
53. MathWorks at http://www.mathworks.com/, MatLab Version 7.5.
54. Moon J. & Moser L. (1965). On cliques in graphs, *Israel Journal of Math*, 3, 23-28.
55. Marcotorchino, J.F. and Michaud, P. (1979). Optimisation en Analyse Ordinale des Donns. Masson, Paris.
56. National Archive of Criminal Justice Data (NACJD) at http://www.icpsr.umich.edu/NACJD/NIBRS/.

57. Neville j., Adler, M., & Jensen, D. (2003). Clustering relational data using attribute and link information, *Proceedings of the eighteenth International Joint Conference on Artificial Intelligence (IJCAI) – Text-Mining and Link Analysis Workshop.*

58. Ng A., Michael J., & Weiss Y. (2001). On spectral clustering: Analysis and an algorithm, *Advances in Neural Information Processing Systems 14* (*NIPS 14).*

59. Pothen, A., Simon, H. D., & Liou, K. P. (1990). Partitioning sparse matrices with eigenvectors of graphs, *SIAM Journal of Matrix Analysis and Application*s, 11, 430–452.

60. Ralambondrainy, H. (1995). A conceptual version of the *k*-means algorithm, *Pattern Recognition Letters,* 16, 1147–1157.

61. Roweis, S. (1997). EM algorithms for PCA and SPCA, *Advances in Neural Information Processing Systems 10 (NIPS 10)*.

62. San, O., Huynh V., & Nakamori, Y. (2004). An alternative extension of the k-means algorithm for clustering categorical data, *International Journal of Applied Mathematics, Computer Science,* 14(2), 241 –247.

63. Selim, S.Z. & Ismail, M.A. (1984). K-Means-type algorithms: A generalized convergence theorem and characterization of local optimality, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 6(1), 81–87.

64. Shannon, C. E. (1948). A Mathematical theory of communication, *Bell System Technical Journal*, 7, 379-423,623-656.

65. Shi, J. & Malik, J. (2000). Normalized cuts and image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 888-905.

66. Skillicorn D. (2004). Applying matrix decomposition to counter-terrorism, external Technical Report ISSN-0836-0227-2004-484, Queen's University, Kingston, Canada.

67. Snyder, H. N. (1999). The over-presentation of juvenile crime proportions in robbery clearance statistics, *Journal of quantitative Criminology*, 13(3), 151-161.

68. Solnim, N & Tishby, N. (1999). Agglomerative information bottleneck, *Advances in Neural Information Processing Systems 12 (NIPS 12)*.

69. Sugar, C. & James, G. (2003). Finding the number of clusters in a data set: An information-theoretic approach, *Journal of the American Statistical Association, 98,750-763.*

70. Suhr, D. (2005). Principal component analysis vs. exploratory factor analysis, *Proceedings of the SAS Users Group International 30 (SUGI 30).*

71. Sumathi, S. & Esakkirajan, S. (2007*). Fundamentals of relational database management systems.* New York: Springer-Verlag.

72. Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic, *Journal of the Royal Statistical Society: Series B*, 63(2), 411–423.

73. Tishby,N., Pereira, F.C., & Bialek, W. (1999). The Information bottleneck method, *37th Annual Allerton Conference on Communication, Control and Computing.*

74. Tomita, E., Tanaka, A., & Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments, *Theoretical Computer Science,* 363, 28–42.

75. Uniform Crime Reporting, National Incident-Based Reporting System, Volume 2 (1992), Data submission Specifications, US Department of Justice, Federal Bureau of Investigation.
76. Uniform Crime Reporting, National Incident-Based Reporting System, Volume 3 (1997), Approaches to implementing an incident-based reporting (IBR) System, US Department of Justice, Federal Bureau of Investigation.
77. Uniform Crime Reporting, National Incident-Based Reporting System, Volume 4 (1999), Error message manual, US Department of Justice, Federal Bureau of Investigation.
78. Uniform Crime Reporting, National Incident-Based Reporting System, Volume 1 (2000), Data Collection Guidelines, US Department of Justice, Federal Bureau of Investigation.
79. Uniform Crime Reporting Handbook, (2004). US department of Justice, Federal Bureau of Investigation.
80. Wagner, D. & Wagner, F. (1993). Between min cut and graph bisection, *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pp 744–750.
81. Wall, M., Rechtsteiner A., & Rocha, L. (2003). Singular value decomposition and principal component analysis, A practical approach to microarray data analysis. D.P. Berrar, W. Dubitzky, M. Granzow, eds. pp 91-109. Norwell, MA: Kluwer.
82. Von Luxburg, U. (2006). A tutorial on Spectral Clustering, Technical Report No. TR-149, Max Planck Institute for Biological Cybernetics.

83. Zaki M., Peters M., Assent I., & Seidl T. (2007). CLICKS: An effective algorithm for mining subspace clusters in categorical data sets, *Data and Knowledge Engineering,* 60, 51-70.

84. Zha, H., Ding, C., Gu, M., & He, X. (2001). Spectral relaxation for K-means clustering, *Advances in Neural Information Processing Systems 14 (NIPS* 14).