ABSTRACT

RACE, SHAINA. Data Clustering via Dimension Reduction and Algorithm Aggregation. (Under the direction of Professor C.D. Meyer).

We focus on the problem of clustering large textual data sets. We present 3 well-known clustering algorithms and suggest enhancements involving dimension reduction. We propose a novel method of algorithm aggregation that allows us to use many clustering algorithms at once to arrive on a single solution. This method helps stave off the inconsistency inherent in most clustering algorithms as they are applied to various data sets. We implement our algorithms on several large benchmark data sets.

Data Clustering via Dimension Reduction and Algorithm Aggregation

by
Shaina Race

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fullfillment of the
requirements for the Degree of
Master of Science

Mathematics

Raleigh, North Carolina

2008

APPROVED BY:

Dr. Ernest Stitzinger	Dr. Ilse Ipsen

Dr. Carl D. Meyer
Chair of Advisory Committee

DEDICATION

This thesis is dedicated to my teachers at Bourne High School who, despite all my protests, gave me the skills I needed to survive as a student.

BIOGRAPHY

Shaina Race was born and raised on Cape Cod, Massachusetts. She obtained a bachelors degree in Mathematical Science from Clemson University in 2006 and a Masters in Applied Mathematics from North Carolina State University in 2008.

ACKNOWLEDGMENTS

The author would like to thank Dr. Carl Meyer, Dr. Amy Langville, and Ralph Abbey for their help in developing the ideas contained in this thesis. She would also like to thank Dimitrios Zeimpekis and Efstratios Gallopoulos, the creators of the MATLAB Text to Matrix Generator (TMG), which was used to parse many of the documents sets used herein. Thanks to Mark Sinka and David Corne from the department of computer science at the University of Reading for creating a benchmark textual dataset for clustering which will be valuable for anyone in this line of research for years to come. We are also grateful for several datasets obtained from the University of California-Irvine Machine Learning Repository set up by David Aha in 1987 and maintained currently by Arthur Asuncion and David Newman.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

0.1 Introduction

Data clustering refers to the process of grouping data based only on information found in the data which describes its characteristics and relationships. Although humans are generally very good at discovering patterns and classifying objects, clustering algorithms are able to discern similarities in data even when humans are not [1]. Clustering has become a foundation of modern data analysis and has numerous applications in a wide variety of fields ranging from social and biological sciences, to business, to statistics, information retrieval, machine learning and data mining.
While clustering has hundreds of applications, this paper is motivated by recent research in the area of document clustering. For this application, our data vectors are documents whose entries indicate the frequencies of terms that they contain. If we let n be the number of documents in our collection and m be the number of terms that appear in our collection, then we create our term-document matrix, $\mathbf{A}$ as follows:

$$A_{m\times n} = \begin{array}{c} \\ \text{Term } 1 \\ \\ \\ \text{Term } i \\ \\ \text{Term } m \end{array} \begin{array}{c} \begin{array}{cccc} \text{Doc } 1 & & \text{Doc } j & \text{Doc } n \end{array} \\ \begin{pmatrix} & & | & \\ & & | & \\ & & | & \\ - & - & f_{ij} & \\ & & & \end{pmatrix} \end{array}$$

Where $f_{i,j}$ is the frequency of term i in document j. Term-document matrices tend to be large and sparse. There are many term-weighting schemes often used for information to downplay the role of frequently used words and lengthy documents. In this paper, we simply normalize the document vectors to have unit 2-norm in order to equate their lengths, since their direction in the term-space is what characterizes them semantically.

The purpose of this paper is to present several common approaches to the problem of clustering and to suggest some alternative algorithms that provide superior results in many cases. There are a number of open issues in clustering. One difficulty is that no single clustering algorithm works best for all problems. Thus, in some respects, you never know what you're going to get. In many scientific contexts, where clusters cannot be evaluated by the human eye, this fact can be discouraging. We propose a method of aggregating several algorithms to obtain an averaging effect, building our confidence in the quality of the resulting clusters. This paper begins with a discussion of well-

known clustering algorithms, and then shows how reducing the dimensionality of data and combining algorithms through aggregation can yield superior results.

0.1.1 *k*-means

k-means is among the oldest, best-known clustering algorithms, and is widely used in applications. It is an iterative procedure that is guaranteed to converge, though not to the best solution. k-means revolves around the placement and replacement of k **centroids**, denoted μ_i. The **centroid** of a cluster is defined to be the average of the vectors in that cluster, i.e. if the vectors $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ form a cluster then the centroid of that cluster is $(1/n)\sum_{i=1}^{n} \mathbf{a}_i$. The k-means algorithm is as follows:

1. Inputs: $m \times n$ matrix of column data, $\mathbf{A}$, and k $m \times 1$ initial centroid vectors, where k is the desired number of clusters. These initial vectors may be chosen at random, but better results are achieved from an initial guess at the centroids of the actual clusters (usually obtained by another clustering algorithm).

2. Each data point is put in the cluster associated with the nearest centroid. If two centroids are equally close, the tie is broken arbitrarily. Any distance metric can be used to determine the distances, in this paper the experiments use the standard Euclidean distance.

3. The new centroid for each cluster is calculated.

4. Repeat steps 2 and 3 until the centroids remain stationary.

5. Output: k clusters where cluster i consists of the data points for which centroid i is the nearest centroid.

The output of k-means is highly sensitive to the initialization of the centroids. Because of this, k-means does not always operate well on its own. It is best combined with an algorithm that can seed it with an initial guess of the centroids of the actual clusters. We use k-means in our dimension reduction algorithms discussed in section 0.2.

0.1.2 The Nonnegative Matrix Factorization (NMF)

The **nonnegative matrix factorization** seeks to decompose an $m \times n$ matrix of column data $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$ into the product of an $m \times r$ matrix of feature vectors, $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_r]$, and a $r \times n$ matrix $\mathbf{H}$ of coefficients. The decomposition is created by solving the following nonlinear optimization problem:

$$\min \|\mathbf{A} - \mathbf{WH}\|_F^2 \text{ such that } \mathbf{W} \geq 0 \text{ and } \mathbf{H} \geq 0$$

The rank of the approximation, r, must be input by the user. The NMF thus provides an additive, parts-based approximation to each data column $\mathbf{a}_j$ in the form of a linear combination of feature vectors as follows:

$$\mathbf{a}_j \approx \sum_{i=1}^{r} \mathbf{h}_{i,j}\mathbf{w}_i \tag{0.1}$$

We can think of this linear combination as a projection of the data onto an r-dimensional "feature space". The coefficients in $\mathbf{H}$ are then (approximately) the coordinates of the data points with respect to the basis for the feature space. Unlike the SVD, NMF does not intend any sort of orthogonality in the feature vectors $\mathbf{w}_i$. There are several NMF algorithms available, the original algorithm was proposed by Lee and Seung in [2] in 1999 and since then several variations and alternatives have been developed. Most NMF algorithms randomly initialize the matrices $\mathbf{W}$ and $\mathbf{H}$ and update them iteratively until convergence. Because of this, NMF solutions are not unique and tend to be sensitive to initializations. The various NMF algorithms themselves are beyond the scope of this paper, but the algorithm used in our experiments, alternating constrained least-squares (ACLS), is described in [3].

Conventional Clustering via NMF

Since the columns in $\mathbf{H}$ represent the coordinates of the data points with respect to the basis feature vectors in $\mathbf{W}$, it is common to cluster data by placing each data point in a cluster associated with a given feature vector. This is done by placing each data point in the cluster associated with the row in which the largest entry in the corresponding column of $\mathbf{H}$ occurs. In other words, we associate the data column $\mathbf{a}_j$ with the feature vector having the largest coefficient in the linear combination from equation (0.1). The algorithm for an $m \times n$ matrix of column data, $\mathbf{A}$ is as follows:

1. Input: Matrix of column data $\mathbf{A}$ and desired number of clusters k.
2. Factor $\mathbf{A} = \mathbf{WH}$ where $\mathbf{W}$ is $m \times k$ and $\mathbf{H}$ is $k \times n$. There are a number of algorithms available for such a factorization, see for example [3].
3. If the maximum value in the column H_j occurs at $H_{i,j}$, place data point A_j into cluster π_i.
4. Output: clusters $\pi_1, \ldots \pi_k$.

The basic NMF clustering algorithm generally provides excellent results and as a result, it is one of the clustering algorithms of choice for the statistical software giant SAS. We will

see from our experiments that superior results can often be obtained by combining NMF with other algorithms, in the same way that k-means is improved when used in conjunction with other algorithms. The NMF algorithm used in the experiments in this paper is called alternating constrained least squares and is described in [3].

0.1.3 Principal Components Analysis (PCA)

The goal of principal components analysis is to reduce a large set of correlated variables to a much smaller set of uncorrelated variables in order to enhance the understanding of the underlying structure of the data [4]. We will avoid the statistical applications of PCA and discuss the scheme for its relevance to clustering. We begin our discussion with some definitions.

We define the **mean** and **variance** for an $m \times n$ matrix of column data, $\mathbf{A} = [\mathbf{a}_1|\mathbf{a}_2|\ldots|\mathbf{a}_n]$, using $\mathbf{e}$ to represent a vector of ones, as follows:

$$\begin{aligned} \mu &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{a}_i = \frac{\mathbf{Ae}}{n} \\ Var[A] &= \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{a}_i - \mu\|_2^2 = \frac{\|\mathbf{A} - \mu\mathbf{e}^T\|_F^2}{n} \\ &= trace\frac{(\mathbf{A} - \mu\mathbf{e}^T)^T(\mathbf{A} - \mu\mathbf{e}^T)}{n} \\ &= \frac{\|\mathbf{A}\|_F^2}{n} - \|\mu\|_2^2 \end{aligned}$$

PCA involves the analysis of eigenvectors of the **centered correlation matrix** $\mathbf{X} = \mathbf{C}^T\mathbf{C}$ where $\mathbf{C} = \mathbf{A} - \mu\mathbf{e}^T/n$. It is well known that these eigenvectors can be recovered from the **singular value decomposition** of $\mathbf{C}$, defined as follows:

Definition 1 (The Singular Value Decomposition (SVD))**.** *For each $\boldsymbol{C} \in \mathbb{R}^{m\times n}$ of rank R, there are orthogonal matrices*

$$\boldsymbol{U}_{m\times m} = [\boldsymbol{u}_1|\boldsymbol{u}_2|...|\boldsymbol{u}_m] \quad \textit{and} \quad \boldsymbol{V}_{n\times n} = [\boldsymbol{v}_1|\boldsymbol{v}_2|...|\boldsymbol{v}_n]$$

and a matrix $\boldsymbol{S}_{R\times R} = diag(\sigma_1, \sigma_2, ..., \sigma_R)$ padded with zeros to make an $m \times n$ rectangular matrix so that

$$\boldsymbol{C} = \boldsymbol{U}\begin{pmatrix} \boldsymbol{S} & 0 \\ 0 & 0 \end{pmatrix}\boldsymbol{V}^T = \sum_{i=1}^{R}\sigma_i\boldsymbol{u}_i\boldsymbol{v}_i^T \quad with \quad \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_R > 0.$$

The σ_i's are the nonzero singular values of $\boldsymbol{C}$, and the respective columns $\boldsymbol{u}_j$ and $\boldsymbol{v}_j$ are the left and right singular vectors of $\boldsymbol{C}$.

The left singular vectors are then eigenvectors for $\mathbf{C}^T\mathbf{C}$ since

$$\mathbf{C}^T\mathbf{C}\mathbf{v}_i = \mathbf{V}\mathbf{S}^2\mathbf{V}^T\mathbf{v}_i = \sigma_i^2\mathbf{v}_i.$$

With the same reasoning. we see that the right singular vectors are eigenvectors for $\mathbf{C}\mathbf{C}^T$.

The first principal component of a data matrix, $\mathbf{A}$, is the normalized direction along which the variance in the columns of $\mathbf{A}$, is maximal. To determine this direction, we want to find the unit vector $\mathbf{x}$ which maximizes the directional variance of $\mathbf{A}$ among all directions in $\mathbb{R}^m$. The orthogonal projection of $\mathbf{a}_j$ onto the direction $\mathbf{x}$ is given by $\widehat{\mathbf{a}_j} = \mathbf{x}\mathbf{x}^T\mathbf{a}_j$, and thus the orthogonal projection of the entire matrix $\mathbf{A}$ is $\widehat{\mathbf{A}} = \mathbf{x}\mathbf{x}^T\mathbf{A}$. It should be clear that the directional variance of the original data matrix $\mathbf{A}$ and the centered data matrix $\mathbf{C}$ are the same along any direction since $\mathbf{C}$ is merely a translation of $\mathbf{A}$. Thus, we want to maximize the variance in $\widehat{\mathbf{C}} = \mathbf{x}\mathbf{x}^T\mathbf{C}$ over all direction vectors $\mathbf{x}$ with $\|\mathbf{x}\|_2 = 1$. This variance is given by

$$Var[\widehat{\mathbf{C}}] = \frac{\|\mathbf{x}\mathbf{x}^T\mathbf{C}\|_F^2}{n} = \frac{\|\mathbf{C}^T\mathbf{x}\|_2^2}{n}.$$

Since

$$\max_{\|x\|_2=1} \|\mathbf{C}^T\mathbf{x}\|_2^2 = \|\mathbf{C}^T\|_2^2 = \sigma_1^2(\mathbf{C})$$

occurs at the first left singular vector of the centered matrix $\mathbf{C}$, $\mathbf{u}_1(\mathbf{C})$ is the first principal component. Hereafter, $\mathbf{u}_1$ is understood to mean $\mathbf{u}_1(\mathbf{C})$. For an illustration of the first principal component of a data cloud (a plot of the columns of $\mathbf{A}$) in $\mathbb{R}^3$ see figure 0.1.

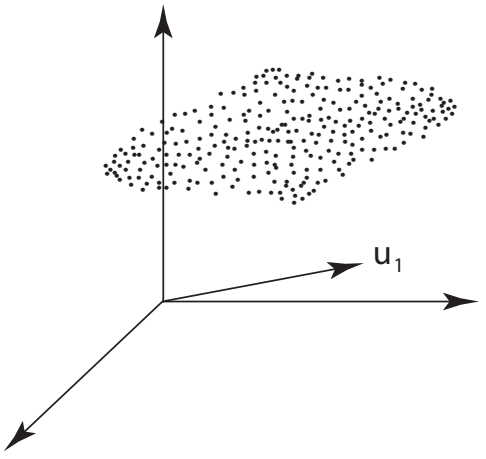

Figure 0.1: Principal Component of a Data Cloud

Principal component analysis involves viewing the m-dimensional data points

along principal components by means of orthogonal projection. Examining the data projected orthogonally onto the direction of the first principal component can reveal some interesting information with regards to clustering. We illustrate this with a simple example.

Consider Figure 0.2 in which a data cloud in $\mathbb{R}^3$ is plotted along with the span of the first principal component. Figure 0.3 depicts the same data cloud orthogonally projected onto the span of the first principal component. The three clusters visible in the original data cloud are clearly visible in their projections onto the span of the first principal component. The true beauty of this is not important in 3-space because we can visualize the clusters before they are projected onto the line. However, when working in higher dimensions, we are unable to visualize the data at all. By reducing the size of the space down to one dimension, oriented along the principal component, we are able to analyze the data with ease.

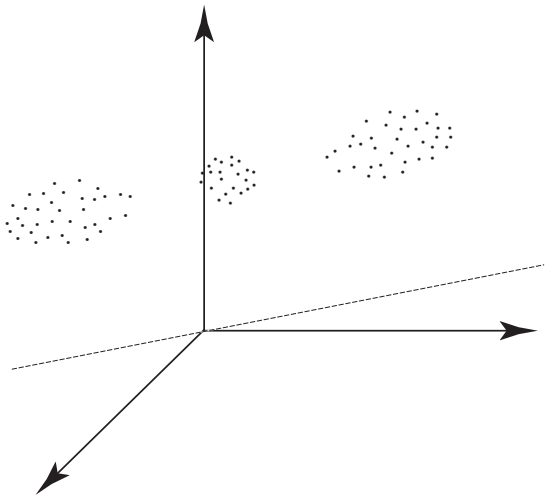

Figure 0.2: Data Cloud with Three clusters

The first principal component does not always give us the information we desire. Consider the situation in Figure 0.4. Here, it is clear that projections onto the first principal component does not reveal the information we desire. This is where other principal components come into the picture.

The **second principal component** is the direction orthogonal to the first principal component along which the variance is maximal. As depicted in Figure 0.5, we define the columns of $\mathbf{P}_{\mathbf{u}_1^\perp}\mathbf{C}$ to represent the projection of the centered data cloud onto $\mathbf{u}_1^\perp$. The same argument used to prove that $\mathbf{u}_1$ points in the direction of maximal variance in $\mathbf{C}$ also shows that $\mathbf{u}_2$ points in the direction of maximal variance in $\mathbf{P}_{\mathbf{u}_1^\perp}\mathbf{C}$.

This development continues further, and the **principal components** of $\mathbf{A}$ are

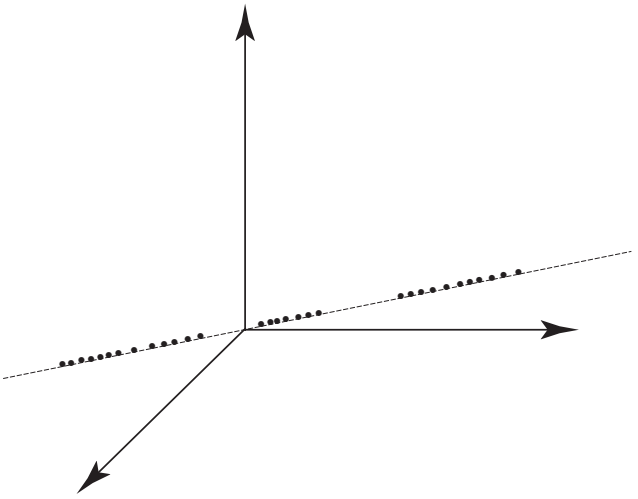

Figure 0.3: Projected Data along the First Principal Component

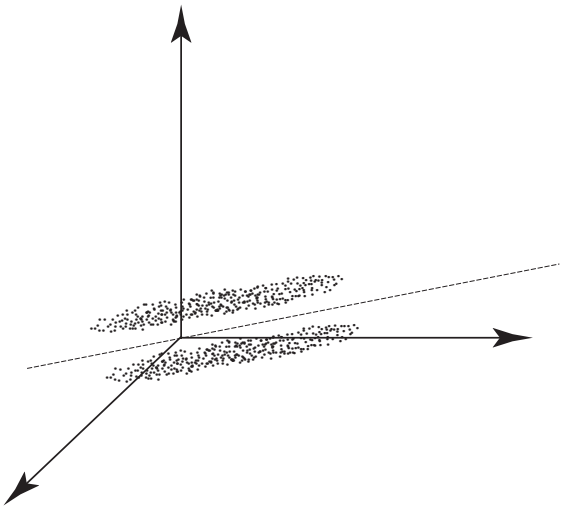

Figure 0.4: Situation where projection onto the first principal component reveals nothing

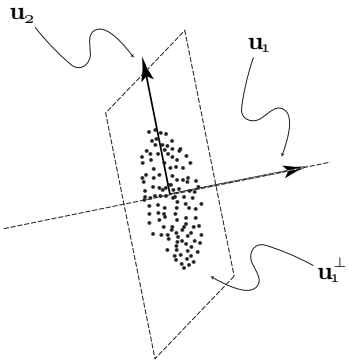

Figure 0.5: Projection of data onto $\mathbf{u}_1^{\perp}$ and direction of the second principal component

thus defined to be the left singular vectors of the centered matrix $\mathbf{C}$. Typically, the analysis involved in PCA involves only the first few principal components. It is especially convenient

that the coordinates of the projected points along the principal components can also be recovered from the SVD, since

$$\mathbf{u}_i^T\mathbf{C} = \mathbf{u}_i^T\mathbf{U}\mathbf{S}\mathbf{V}^T = \sigma_1\mathbf{v}_i^T. \tag{0.2}$$

Now that we have developed the underlying geometry of PCA, we see how its relevance to clustering has been applied algorithmically.

0.1.4 Principal Direction Divisive Partitioning (PDDP)

Principal direction divisive partitioning (PDDP) is an iterative algorithm proposed in [5] by Daniel Boley at the University of Minnesota. In this scheme, the centered data are projected orthogonally onto the span of the first principal component and divided into 2 groups based upon whether these projections of the centered data vectors, $\mathbf{c}_i$, fall to the left or the right of zero (hence whether the original data points $\mathbf{a}_j$ fall to the left or right of the mean). Once the data are divided, the two groups are examined to find the one with the greatest variance (scatter). This subset of data is then extracted from the original data matrix, centered and projected onto the span of its own first principal component. The split at zero is made again and the algorithm proceeds iteratively until the desired number of clusters has been produced.

In light of equation (0.2), to split the data at each iteration of the PDDP algorithm, one only has to divide those data points whose corresponding entry in $\mathbf{v}_1$ is positive from those whose corresponding entry is negative. Data points whose corresponding entries are zero are arbitrarily put in either cluster. More precisely, the algorithm is as follows:

1. Input: Matrix of column data $\mathbf{A}$ and desired number of clusters k.
2. Compute the singular value decomposition of the centered matrix

   $$\mathbf{C} = \mathbf{A} - \mu\mathbf{e}^T = \mathbf{A}(\mathbf{I} - \mathbf{e}\mathbf{e}^T/n)$$

   as $\mathbf{C} = \mathbf{U}\mathbf{S}\mathbf{V}^T$. Let $\mathbf{v}_1$ be the first column of $\mathbf{V}$ (i.e. the first row of $\mathbf{V}^T$)
3. Divide the data into two clusters, $\pi_1 = \{i|\mathbf{v}_{1,i} \leq 0\}$ and $\pi_2 = \{i|\mathbf{v}_{1,i} > 0\}$.
4. Compare the variance of the existing clusters. Extract the columns of $\mathbf{A}$ corresponding to the cluster with the largest variance. Use this subset of data as the data matrix for the next iteration.

5. Repeat steps 2-4 until the desired number of clusters is obtained

0.2 Clustering via Dimension Reduction

One of the two main themes of this paper is clustering data by means of dimension reduction. It is apparent through experimentation that it is difficult for classical clustering algorithms to determine clusters in high-dimensional data. This fact is not surprising given the chaotic nature of large data sets. We propose reducing the dimensions of data so that the inherent clusters become clearer. The motivation for this idea begins with a concept from early information retrieval, called Latent Semantic Indexing (LSI).

0.2.1 Latent Semantic Indexing (LSI)

Latent Semantic Indexing was first proposed in [6] and quickly grew roots in the information retrieval community. The approach seeks to deal with the randomness of word choice by relating terms and documents that are closely associated semantically. For example, a human would not differentiate between the words "car" and "automobile", but computers see the words as two separate entities. LSI seeks to use the associative patterns of these two words in a document set, i.e. the fact that the two words are used in similar contexts, to uncover their latent relationship. The authors in [6] suggest the use of the singular value decomposition to project the documents into a lower-dimensional space which reflects the major associative patterns of the data while ignoring less important influences. This is done with a simple truncation of the singular value decomposition as follows:

Let $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$ be our matrix of column data, normalized so that each column has unit 2-norm. It is well known that the truncated SVD provides the closest rank r approximation to the original data matrix in the 2-norm. More specifically, if the SVD of $\mathbf{A}$ is $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ and we let $\mathbf{U}_r$ be the first r columns of $\mathbf{U}$ and $\mathbf{V}_r^T$ be the first r rows of $\mathbf{V}^T$ and $\mathbf{S}_r = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ then the closest rank r approximation of $\mathbf{A}$ is

$$\mathbf{A} \approx \mathbf{U}_r\mathbf{S}_r\mathbf{V}_r^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

By the very nature of dimension reduction, it becomes possible for two documents with similar semantic properties to be mapped closer together. Unlike the NMF, LSI makes no attempt to interpret the terms $\mathbf{u}_i\mathbf{v}_i^T$, instead, as stated in [6], the aim is "merely to be able to represent terms, documents, and queries in a way that escapes the unreliability, ambiguity and redundancy of individual terms as descriptors." We take advantage of LSI

for clustering in our SVD-r algorithm.

0.2.2 SVD-r Algorithm

Let $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$ be our matrix of column data, and let $\mathbf{C}$ be the centered data matrix $\mathbf{A} - \mu\mathbf{e}^T/n$. As stated above, the closest rank r approximation of $\mathbf{C}$ is

$$\mathbf{C} \approx \mathbf{U}_r\mathbf{S}_r\mathbf{V}_r = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

Furthermore, as shown in equation (0.2), the entries in the left singular vectors, $\mathbf{v}_1, \ldots, \mathbf{v}_r$ provide us with the coordinates of the centered data projected orthogonally onto the first r principal components. By projecting the data onto the space provided by the principal components, we reduce the dimension of our problem in a way that eases the task of clustering.

The proposed algorithm (SVD-r) is as follows:

1. Input: Desired number of clusters, k, and desired dimension reduction, r
2. Calculate the singular value decomposition of the centered matrix, $\mathbf{C} = \mathbf{USV}^T$.
3. Let $\mathbf{Z}$ be the matrix consisting of the first r columns of $\mathbf{V}$, $\mathbf{Z} = [\mathbf{v}_1, \ldots, \mathbf{v}_r]$
4. Let $\mathbf{y}_i$ be the i^{th} row of $\mathbf{Z}$
5. Cluster the r-dimensional data points $\mathbf{y}_i$ into k clusters, $C_1, \ldots, C_k$, using any desired clustering algorithm.
6. Output: Clusters $\pi_1, \ldots \pi_n$ where $\pi_i = \{j | y_j \in C_i\}$

We will use SVD-k to denote the above algorithm when $r = k$ is the number of desired clusters.

Centered vs. Uncentered

It has been realized experimentally that good (often superior) results can be obtained without centering the data. In this situation, we can no longer view the singular vectors as principal components. Instead, the first singular vector represents the direction of the line, $\mathcal{T}$, through the origin for which the total sum of squares of orthogonal deviations between the column data in $\mathbf{A}$ and $\mathcal{T}$ is minimal among all zero-intercept lines in $\mathbb{R}^m$. We formalize this with the following theorem:

Theorem 1 (Zero-Intercept Total Least Squares)**.** *The line* $\mathcal{T} = \{\alpha \boldsymbol{u}_1(\boldsymbol{A}) | \alpha \in \mathbb{R}\}$ *is the line which minimizes the sum of the squared orthogonal deviation of the columns in* $\boldsymbol{A}$ *among all zero-intercept lines in* $\mathbb{R}^m$

Proof. Let $\mathbf{x} \in \mathbb{R}^m$ be any vector with $\|\mathbf{x}\|_2 = 1$. As depicted in figure 0.6, the orthogonal projection, $\widehat{\mathbf{A}_j}$ of the j^{th} column, $\mathbf{a}_j$, of $\mathbf{A}$ onto the line $\mathcal{T} = \{\alpha \mathbf{x} | \alpha \in \mathbb{R}\}$ is $\widehat{\mathbf{a}_j} = \mathbf{x}\mathbf{x}^T\mathbf{a}_j$.

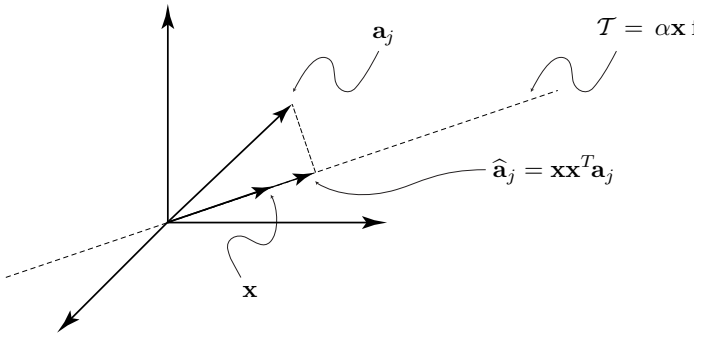

Figure 0.6: Deviation between $\mathbf{a}_j$ and a line through the origin

The total sum of squared orthogonal deviations is

$$\delta(\mathbf{x}) = \sum_{j=1}^{n} \|(I - \mathbf{x}\mathbf{x}^T)\mathbf{a}_j\|_2^2 = \|(I - \mathbf{x}\mathbf{x}^T)\mathbf{A}\|_F^2 = trace(\mathbf{A}^T\mathbf{A}) - trace(\mathbf{A}^T\mathbf{x}\mathbf{x}^T\mathbf{A}) = \|\mathbf{A}\|_F^2 - \|\mathbf{A}^T\mathbf{x}\|_2^2$$

Since $\delta(\mathbf{x})$ attains its minimum value at points where $\|\mathbf{A}^T\mathbf{x}\|_2$ is maximized, and since the maximum value

$$\max_{\|x\|_2=1} \|\mathbf{A}^T\mathbf{x}\|_2 = \|\mathbf{A}^T\|_2 = \sigma_1(\mathbf{A})$$

occurs at $\mathbf{x} = \mathbf{u}_1(A)$, the result is established. □

Now that the interpretation of $\mathbf{u}_1(\mathbf{A})$ is clear, it is easy to see that the relationship between the singular vectors of $\mathbf{C}$ and $\mathbf{A}$ is virtually nonexistent. The directions of the two vectors could be anything from orthogonal to coincident.

However, by the virtue of dimension reduction, it is possible for good clustering results to stem from this uncentered scenario. Depending on the location and separation of the clusters in m-space, projection onto the zero-intercept total least squares line may still reveal the information desired. It is clear through experimentation that the uncentered case should not be overlooked, and for this reason we include the uncentered SVD-r algorithm in our experiments, and encourage future work to determine when exactly this is best.

0.2.3 NMF-r Algorithm

Another group of dimension-reduction clustering algorithms are born from the nonnegative matrix factorization (NMF), which also provides us with an approximation to the original data matrix in a lower dimensional space. Rather than clustering data into groups according to the dominant feature vector associated with each point, we suggest clustering the data according to their coordinates in the lower dimensional "feature space" provided by NMF. In doing so, we allow more features in data than there are clusters (unlike the conventional NMF clustering algorithm, which requires the number of features found by the decomposition to be equal to the number of clusters desired). The basic proposed algorithm is quite simple:

1. Input: Matrix of column data $\mathbf{A}$, desired dimension of feature space, r, and desired number of clusters, k.

2. Factor $\mathbf{A} \approx \mathbf{WH}$ where $\mathbf{W}$ is $m \times r$ and $\mathbf{H}$ is $r \times n$.

3. Cluster the columns of $\mathbf{H}$, which represent the coordinates of our data points after projection into the r-dimensional feature space, into k clusters. Any clustering algorithm can be used here.

4. Output: clusters determined by clustering algorithm in step 3.

0.3 Cluster Aggregation

The other main theme of this paper is that of cluster aggregation. Cluster aggregation is a process which allows us to combine results from various clustering algorithms in order to get a better picture of what our clusters should actually look like. It is based upon the assumption that assignment errors made by clustering algorithms will be outnumbered by correct assignments, and that although any clustering algorithm is bound to make errors, rarely will different algorithms make the *same* error. The process is really quite simple. We create a similarity matrix, called the **aggregation matrix** as follows: Let $\Pi_1, \ldots, \Pi_r$ be the cluster results of r different clustering algorithms. We define the $n \times n$ aggregation matrix $\mathbf{M}$ (where n is the number of original data points) so that $\mathbf{M}_{i,j} =$ the number of times data point i has been clustered with data point j in $\Pi_1, \ldots, \Pi_r$. The maximum entry for $\mathbf{M}_{i,j}$ is thus r, indicating that x_i and x_j were clustered together by every algorithm used. This similarity matrix $\mathbf{M}$ can then be clustered once more using any desired clustering algorithm. This process of cluster aggregation not only allows us to combine clustering results from different algorithms, but also allows us to iterate randomly

initialized algorithms such as k-means and the basic NMF clustering algorithm to incorporate the results of different runs into one matrix that can once again be analyzed for dominant clusters. If we define an **adjacency matrix of clustering** from each algorithm to be the matrix whose $(i, j)^{th}$ entry is 1 if data point i and data point j were put in the same cluster, then the aggregation matrix is simply the sum of the adjacency matrices of clustering produced by the desired algorithms. This scheme is illustrated in Figure 0.7.

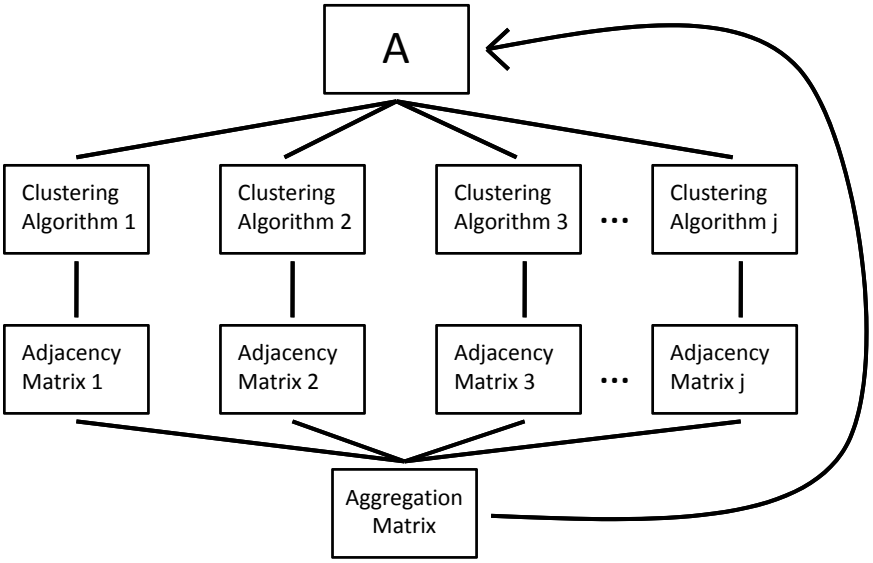

Figure 0.7: Algorithm Aggregation

Since the aggregated matrix can be clustered in any number of ways, it should be clear that the process can be repeated iteratively. We can run several algorithms on the aggregated matrix and again aggregate these results into a second matrix for clustering. This idea is demonstrated in our experiments, showing that iterated aggregation tends to yield a common solution among many algorithms. Although this common solution may not have the highest accuracy of all the algorithms, in uncertain situations where the "answer" to the clustering is unable to be evaluated by a human, we are able to have confidence that the solution is at least as good (generally much better) than the average solution.

Another action that may improve the clustering results on the aggregation matrix

is dropping all values below a certain tolerance down to zero. For example, if you ran ten clustering algorithms, aggregated the results and found that two data points were only clustered together by two algorithms, you might make a reasonable assumption that those two pairings were made in error. Dropping these low valued entries in the aggregation matrix filters noise from the aggregation data, yielding superior results.

0.4 Cluster Evaluation

Cluster evaluation, or cluster validation as it is more traditionally known, is an important aspect of any cluster related research. Since most existing clustering algorithms will determine clusters in data whether or not they exist naturally, it is important to have some way to evaluate the accuracy of clustering results. Cluster evaluation measures are typically broken into two catagories, internal (or unsupervised), and external (or supervised). Internal measures use no outside information, such as class or catagory labels, to determine the validity of the clustering. Internal measures are typically measures of cluster cohesion and separation. Cluster cohesion gives us an idea of how close together elements in an individual cluster are while cluster separation tells us how distinct or separated the clusters are from each other. Since internal measures do not tell us explicitly about the accuracy of our clustering results, we do not use them in this paper, however we suggest their use to help determine that the clustering results from the aggregation process agree.

External measures use information not included in the dataset (such as categorical labels which provide the "answer" to the clustering) to determine how well the algorithm clustered the data into their pre-determined categories. External measures are not useful in practice because there is no need to cluster data which are already catagorically assigned, but they give us a more accurate metric for comparing different clustering algorithms. In this paper we will simply compute the **accuracy** of the clustering as the percentage of documents (or data points) that were clustered correctly. Although this metric is simple in concept, the actual calculation involves some clever graph theoretical techniques.

0.4.1 Accuracy as a Cluster Metric

Like any external measure, the accuracy metric requires the input of predetermined class labels for the data. Once the documents are clustered using a clustering algorithm, they acquire a different set of labels, associated with the clusters to which they were assigned. For the sake of discussion we will call the predetermined class labels "an-

swer labels" and the assigned cluster labels "cluster labels". We first create an $m \times m$ **confusion matrix**, **M**, which tells us how many data points from each answer label were given each cluster label. More specifically, the confusion matrix is defined as follows:

$$\mathbf{M}_{i,j} = \text{number of documents with answer label } i \text{ assigned cluster label } j$$

Take for example the following confusion matrix for a data set with 3 clusters:

$$\begin{array}{c} \\ \text{answer 1} \\ \text{answer 2} \\ \text{answer 3} \end{array} \begin{array}{c} \begin{array}{ccc} \text{cluster 1} & \text{cluster 2} & \text{cluster 3} \end{array} \\ \begin{pmatrix} 9 & 8 & 2 \\ 7 & 1 & 2 \\ 0 & 10 & 16 \end{pmatrix} \end{array} \tag{0.3}$$

From this matrix, we can read that 8 documents with answer label 1 were assigned the cluster label 2, whereas 7 documents with answer label 2 were assigned the cluster label 1. When trying to decide which documents were correctly classified, we see why someone decided to call this the confusion matrix. A clustering algorithm is 100 % accurate if it divides the data into the predetermined classes, however, we do not expect it to label the clusters in the same order that we did. Thus, we are left to determine which cluster label was intended by the algorithm to represent which answer label. For the small example in (0.3), using the process of elimination, we can pick out the best assignment of cluster labels to answer labels. We would naturally select answer 3 to be associated with cluster 3. Then, upon gauging the possibilities, it seems natural that answer 2 be associated with cluster 1, leaving answer 1 associated with cluster 2. This would yield a total of $16 + 7 + 8$ correctly classified documents out of 55, an accuracy of $\frac{31}{55} = 56\%$.

The example above outlines a more general assignment problem that is commonly known in graph theory as a maximum matching. Creating a bipartite graph in which one side consists of answer labels and one side consists of cluster labels, and using the confusion matrix as an adjacency matrix for this graph, we seek to choose a set of edges that match each cluster label with a unique answer label in a way that maximizes the sums of the weights of the edges. There are several ways to solve this problem, the most famous of which is the Kuhn-Munkres algorithm, also known as the Hungarian Algorithm. The details of this algorithm are beyond the scope of this paper; further information can be found in [7] and [8].

Once each cluster label is associated with an answer label, we can then count the number

of incorrectly labeled documents and compute the accuracy of the clustering.

0.5 Results and Discussion

The first column of accuracies in the following tables represent the algorithms run on the actual matrix of column data **A**, with the last 4 rows indicating the algorithms were run on the matrix **H** from the the corresponding r-dimensional NMF factorization. The algorithm abbreviations in the tables are as follows:

1. **NMF-Basic:** Basic NMF clustering algorithm as discussed in section 0.1.2.
2. **PDDP:** Principal Direction Divisive Partitioning as discussed in section 0.1.4.
3. **PDDP-kmeans:** PDDP clusters seeded to k-means .
4. **SVDr-PDDP-kmeans:** centered SVD-r clustering using PDDP seeding k-means
5. **un.SVDr-PDDP-kmeans:** uncentered SVD-r clustering using PDDP seeding k-means
6. **H-PDDP** PDDP run on **H** from the r-dimensional NMF factorization.
7. **H-PDDP-kmeans** PDDP seeding k-means run on **H** from the r-dimensional NMF factorization.
8. **H-SVDk-PDDP-kmeans** centered SVD-k algorithm run on **H** from the r-dimensional NMF factorization, using PDDP seeding k-means .
9. **H-un.SVDk-PDDP-kmeans** uncentered SVD-k algorithm run on **H** from the r-dimensional NMF factorization, using PDDP seeding k-means .

For algorithms 8 and 9 a two-fold dimension reduction is taking place. The data matrix is first factored into its nonnegative components, **W** and **H**, which reduces the dimension from m to r. When we run the SVD-k algorithm on H, we are again reducing the dimension from r to k.

0.5.1 Dataset: Medlars/Cranfield/CISI

This dataset is composed of the Medlars, Cranfield, and CISI datasets (available from Michael Berry's LSI page at http://www.cs.utk.edu/ lsi). We used the pre-existing term-document matrices created by Zeimpekis and Gallopoulos's Term-Matrix Generator

(TMG) (available from http://scgroup6.ceid.upatras.gr:8000/wiki/index.php/Main_Page) combined these 3 matrices and set our clustering algorithms to re-separate the documents into 3 clusters. The columns of the data matrix were normalized so each had unit 2-norm.

Experiment 1

The goal of this experiment was to investigate the behavior of iterated aggregation from the clusters given by the various algorithms for a single value of r. No clustering results are found by the basic NMF clustering algorithm when $r \neq k$, and hence the empty set is displayed for the corresponding accuracies. In the second columns of these tables, the algorithms were run on the data matrix using the indicated value of r. The third column, labeled "aggregated", contains the accuracies of each algorithm run on the aggregation matrix formed from the clusters found in the second column. The forth column is"double aggregation" where the results from the third column were aggregated once again and clustered by each algorithm. The fifth and subsequent columns follow suit, aggregating the results from the previous column and clustering that aggregation matrix. For all of the aggregation columns, we required that two documents be clustered together at least 5 times in order for their relationship to be recorded in the aggregation matrix. In other words, the minimum entry in the aggregation matrix was set to 5.

As you can see from the following tables, the number of aggregation iterations required can vary. The boxed entries demonstrate the "convergence" of the aggregation results, indicating that several of the algorithms are agreeing upon the same clusters. In table 0.1 with $r = 3$, almost every algorithm agrees on the same clustering after triple aggregation. Notice that the accuracy of this agreed-upon clustering is relatively high compared to the accuracies of the algorithms alone in the second column. In table 0.5.1, four algorithms found the same clustering after the first aggregation, and in table 0.3 it again took 3 iterations for any noticeable agreement to take place. The extent to which the algorithms should agree before terminating or altering the process will have to be determined by the user based on the dataset and problem at hand. Notice that as r increases, so do the accuracies of the clusterings, however increasing r too high adds noise and diminishes the accuracy.

Experiment 2

The purpose of this experiment was to use our aggregation technique with more clusterings as input. Here we used multiple values of r to produce results similar to

Table 0.1: Experiment 1: Cluster Accuracies for Med/Cran/CISI

Algorithm	r=k=3	Aggregated	Double	Triple
NMF Basic	0.70084811	0.89771267	0.9306091	0.89719866
PDDP	0.83012079	0.89437163	0.89437163	0.89719866
PDDP-kmeans	0.96376253	0.89282961	0.89282961	0.89719866
SVDr-PDDP-kmeans	0.81624261	0.72834747	0.769211	0.89719866
un.SVDr-PDDP-kmeans	0.75970188	0.89719866	0.89719866	0.89719866
H-PDDP	0.59650475	0.89437163	0.89874068	0.89719866
H-PDDP-kmeans	0.71626831	0.89719866	0.89719866	0.89719866
H-SVDk-PDDP-kmeans	0.8234387	0.89334361	0.89334361	0.9308661
H-un.SVDk-PDDP-kmeans	0.71446929	0.9308661	0.9308661	0.89719866

Table 0.2: Experiment 1: Cluster Accuracies for Med/Cran/CISI

Algorithm	r=6	Aggregated
NMF Basic	[]	0.91673092
PDDP	0.83012079	0.9164739
PDDP-kmeans	0.96376253	0.91673092
SVDr-PDDP-kmeans	0.90105371	0.79362632
un.SVDr-PDDP-kmeans	0.83037779	0.79105628
H-PDDP	0.88923156	0.91647391
H-PDDP-kmeans	0.94937034	0.91673092
H-SVDk-PDDP-kmeans	0.8694423	0.91390388
H-un.SVDk-PDDP-kmeans	0.692624	0.91673092

the first columns of the tables in experiment 1, and used these results for aggregation. The resulting accuracies are documented in table 0.4. After running our algorithms with $r = 3, 6, 9,$ and 12 we had a total of 33 clusterings. We then dropped all values from the aggregation matrix which were less than or equal to 14. On the second iteration of aggregation, 4 of the algorithms agreed upon a common clustering. While k-means seeded by PDDP provides excellent results on this dataset, the next experiments show us why we can't have confidence that one algorithm will consistently outperform the others in any given situation.

Table 0.3: Experiment 1: Cluster Accuracies for Med/Cran/CISI

Algorithm	r=15	Aggregated	Double	Triple
NMF Basic	[]	0.96530455	0.95334439	0.96453354
PDDP	0.83012079	0.94962735	0.94962735	0.96376253
PDDP-kmeans	0.96376253	0.96530455	0.96530455	0.96453354
SVDr-PDDP-kmeans	0.94500129	0.71652531	0.61783603	0.73477255
un.SVDr-PDDP-kmeans	0.78026214	0.61578001	0.92572603	0.83320483
H-PDDP	0.86096119	0.94962735	0.97044462	0.96376253
H-PDDP-kmeans	0.97584169	0.96530455	0.96556155	0.96453354
H-SVDk-PDDP-kmeans	0.86533025	0.96067849	0.95451041	0.96530455
H-un.SVDk-PDDP-kmeans	0.60164482	0.96530455	0.96453354	0.96453354

Table 0.4: Experiment 2: Cluster Accuracies for Med/Cran/CISI

Algorithm	r=k=3	r=6	r=9	r=12	Aggregated	Double
NMF Basic	0.7026	[]	[]	[]	0.9365	0.7219
PDDP	0.8301	0.8301	0.8301	0.8301	0.926	0.9363
PDDP-kmeans	0.9638	0.9638	0.9638	0.9638	0.9432	0.9363
SVDr-PDDP-kmeans	0.8165	0.9011	0.9414	0.9435	0.9288	0.6214
un.SVDr-PDDP-kmeans	0.7597	0.8304	0.79	0.8258	0.9365	0.9363
H-PDDP	0.5965	0.8892	0.8808	0.5759	0.9609	0.9365
H-PDDP-kmeans	0.7158	0.9494	0.9686	0.5477	0.9365	0.9291
H-SVDk-PDDP-kmeans	0.825	0.8692	0.5723	0.861	0.9314	0.9363
H-un.SVDk-PDDP-kmeans	0.7155	0.6926	0.7289	0.604	0.9365	0.9429

0.5.2 Benchmark Dataset created by Sinka and Corne

Mark Sinka and David Corne from the department of computer science at the University of Reading proposed a large benchmark dataset for document clustering in [9]. The set contains 11,000 documents extracted from the web, 100 documents on each of 11 different topics. The set contains 4 topics, banking/finance, programming, science, and sport, and each topic contains 2 or 3 subcategories: commerical banks, building societies, insurance agencies, java, C/C++, visual basic, astronomy, biology, soccer, motor sport, and sport. This dataset is extremely noisy. Many of the documents appear to be nothing more than a list of words, others contain nothing but addresses and links. Some documents are very long, containing comment postings from blogs and consumer sites. Other documents are less than 5 words in length. For more information on how the dataset was engineered and how to obtain the dataset, see [9]. The original documents were in html format and for these experiments we simply removed all the html and javascript tags and parsed the text that remained. We used Zeimpekis and Gallopoulos's TMG [10] to parse the documents and applied stemming and normalization.

Experiment 3: Groups B,C,F, and G combined

For these experiments, we combined the 4000 documents from sets B,C,F, and G which correspond to the following topics (and categories): Banking/Finance (Building Societies), Banking/Finance (Insurance Agencies), Programming Languages (Visual Basic), Science (Astronomy). We set the minimum global frequency to 4, so words appearing less than 4 times in the first collection were discarded from the dictionary. The tables have the same format as those in experiment 1. We dropped entries in the aggregation matrix that were less than or equal to 5. These experiments demonstrate the effectiveness of the NMF and SVD-k algorithms proposed in this paper. They also reflect the fact that no algorithm can be trusted to find a good solution on all datasets. Take note of the difference in the

performance of PDDP on this dataset compared to table 0.1. Also note the increase in the performance of the proposed NMF algorithms as the number of feature vectors, r, increases.

Table 0.5: Experiment 3: Cluster Accuracies for BenchmarkBCFG

Algorithm	r=4	Aggregated	Double	Triple
NMF Basic	0.62725	0.69275	0.57725	0.69425
PDDP	0.4505	0.67775	0.69325	0.69425
PDDP-kmeans	0.34025	0.69325	0.69375	0.69425
SVDr-PDDP-kmeans	0.69825	0.549	0.51575	0.69425
un.SVDr-PDDP-kmeans	0.74725	0.69475	0.6945	0.69425
H-PDDP	0.582	0.67775	0.5665	0.69425
H-PDDP-kmeans	0.65775	0.69275	0.5745	0.69425
H-SVDk-PDDP-kmeans	0.66125	0.67075	0.695	0.69425
H-un.SVDk-PDDP-kmeans	0.6825	0.69475	0.57825	0.69425

Table 0.6: Experiment 3: Cluster Accuracies for BenchmarkBCFG

Algorithm	r=10	Aggregated	Double	Triple	Quadruple	Quintuple
NMF Basic		0.74725	0.75225	0.58025	0.427	0.74775
PDDP	0.4505	0.74275	0.75125	0.74775	0.74775	0.74775
PDDP-kmeans	0.34025	0.75275	0.748	0.75225	0.75225	0.74775
SVDr-PDDP-kmeans	0.71975	0.5945	0.49725	0.6395	0.564	0.72225
un.SVDr-PDDP-kmeans	0.67325	0.703	0.51575	0.52225	0.538	0.63225
H-PDDP	0.71	0.74275	0.752	0.74775	0.60225	0.74775
H-PDDP-kmeans	0.7635	0.74725	0.75225	0.58025	0.44225	0.74775
H-SVDk-PDDP-kmeans	0.7255	0.736	0.748	0.7455	0.6055	0.76525
H-un.SVDk-PDDP-kmeans	0.788	0.74975	0.748	0.58	0.4265	0.74775

Experiment 4: Groups G,H, and I combined

For these experiments we used the two subsets in the science topic, pertaining to the subcategories of Astronomy and Biology, and one subset from the sport topic, pertaining to soccer. In table 0.7 we set the algorithms to find 2 clusters, hoping they could group the scientific topics together. The minimum global frequency was set at 2. Normalization and stemming were used. Entries less than or equal to 4 were dropped from the aggregation matrix. The aggregation technique is able to more or less decide on a cluster solution after 1 iteration. The accuracies of the aggregated solutions with $r = 2$ and $r = 6$ are about equal.

Next, we set the algorithms to find 3 clusters, in an attempt to separate the science topic into the subcategories biology and astronomy.

Table 0.7: Experiment 4: Cluster Accuracies for BenchmarkGHI - $k = 2$ Clusters

Algorithm	r=2	Aggregated
NMF Basic	0.984	0.982
PDDP	0.96833333	0.982
PDDP-kmeans	0.97966667	0.982
SVDr-PDDP-kmeans	0.97733333	0.977
un.SVDr-PDDP-kmeans	0.98433333	0.982
H-PDDP	0.979	0.982
H-PDDP-kmeans	0.984	0.982
H-SVD-PDDP-kmeans	0.88333333	0.977
H-un.SVD-PDDP-kmeans	0.98366667	0.982

Table 0.8: Experiment 4: Cluster Accuracies for BenchmarkGHI - $k = 2$ Clusters

Algorithm	r=6	Aggregated
NMF Basic	[]	0.981
PDDP	0.96833333	0.98
PDDP-kmeans	0.97966667	0.98033333
SVDr-PDDP-kmeans	0.98933333	0.96966667
un.SVDr-PDDP-kmeans	0.987	0.97766667
H-PDDP	0.975	0.98
H-PDDP-kmeans	0.981	0.981
H-SVD-PDDP-kmeans	0.97966667	0.98
H-un.SVD-PDDP-kmeans	0.98366667	0.98

Table 0.9: Experiment 4: Cluster Accuracies for BenchmarkGHI - $k = 3$ Clusters

Algorithm	r=k=3	Aggregated
NMF Basic	0.89333333	0.661
PDDP	0.88666667	0.894666666666667
PDDP-kmeans	0.90533333	0.894666666666667
SVDr-PDDP-kmeans	0.765	0.65833333
un.SVDr-PDDP-kmeans	0.891	0.894666666666667
H-PDDP	0.889	0.894666666666667
H-PDDP-kmeans	0.89333333	0.661
H-SVD-PDDP-kmeans	0.891	0.89433333
H-un.SVD-PDDP-kmeans	0.892	0.66066667

Table 0.10: Experiment 4: Cluster Accuracies for BenchmarkGHI - $k = 3$ Clusters

Algorithm	r=8	Aggregated
NMF Basic		0.899
PDDP	0.88666667	0.89866667
PDDP-kmeans	0.90533333	0.899
SVDr-PDDP-kmeans	0.723	0.90066667
un.SVDr-PDDP-kmeans	0.79833333	0.47766667
H-PDDP	0.88	0.89866667
H-PDDP-kmeans	0.82033333	0.899
H-SVD-PDDP-kmeans	0.876	0.898
H-un.SVD-PDDP-kmeans	0.898	0.899

Experiment 5: Groups A,B,F, and J combined

For this experiment we combined 1000 documents from each of 4 categories. The columns of the resulting term-document matrix were normalized and we set our algorithms to find the 4 clusters. We used each algorithm with the 4 indicated values for r for a total

of 37 clusterings. We then aggregated these results, dropping entries in our aggregation matrix less than or equal to 10. Entries less than or equal to 2 were dropped in the double aggregation.

Table 0.11: Experiment 5: Cluster Accuracies for benchmarkABFG

Algorithm	r=4	r=8	r=12	r=18	Aggregated	Double
NMF Basic	0.6518				0.714	0.7558
PDDP	0.7208	0.7208	0.7208	0.7208	0.7542	0.7558
PDDP-kmeans	0.7852	0.7852	0.7852	0.7852	0.756	0.7558
SVDr-PDDP-kmeans	0.6708	0.721	0.7215	0.7858	0.7615	0.7602
un.SVDr-PDDP-kmeans	0.678	0.7808	0.8322	0.7108	0.759	0.759
H-PDDP	0.7018	0.7508	0.748	0.7448	0.7582	0.7622
H-PDDP-kmeans	0.671	0.8048	0.7855	0.7922	0.7235	0.7558
H-SVD-PDDP-kmeans	0.684	0.7515	0.6882	0.7335	0.7632	0.759
H-un.SVD-PDDP-kmeans	0.669	0.7392	0.7338	0.7308	0.6102	0.7558

0.5.3 Iris Data Set

This data set is not textual, it is scientific. It was obtained from [11] and contains information on 150 flowers. Each flower was measured with four characteristics: sepal length, sepal width, petal length, and petal width. Of these flowers there are 3 different species. We removed the species variable from the set and clustered the data into $k = 3$ clusters, hoping to separate the flowers into categories according to species. No scaling or normalization was used.

Experiment 6

The table below indicates that the algorithms were run with $r = 3$ and $r = 4$. These results were then combined in an aggregation matrix, dropping values less than or equal to 5. Six of the nine algorithms agree on a clustering after one aggregation iteration. More importantly, the clustering they agree on matches the highest accuracy obtained from the original algorithms.

Table 0.12: Experiment 6: Cluster Accuracies for Iris Data

Algorithm	r=3	r=4	Aggregated
NMF	0.6733		0.6667
PDDP	0.9733	0.9733	0.9733
PDDP-kmeans	0.9733	0.9733	0.9733
SVDr-PDDP-kmeans	0.7067	0.9533	0.9733
un.SVDr-PDDP-kmeans	0.78	0.8	0.9733
H-PDDP	0.6467	0.9667	0.9733
H-PDDP-kmeans	0.9467	0.98	0.7933
H-SVDk-PDDP-kmeans	0.9467	0.9733	0.9733
H-un.SVDk-PDDP-kmeans	0.7133	0.7533	0.72

0.6 Conclusions

We have discussed 3 commonly used document clustering algorithms, and proposed a technique of dimension reduction to increase the accuracy of those algorithms. The dimension reduction technique is not limited to the algorithms discussed in this paper, but can be implemented with any clustering algorithm the user desires.

One of the main difficulties in data clustering is the inconsistency of any algorithm when applied to various data sets. There is no clustering algorithm that will out-perform all others on all data sets [12]. Cluster Aggregation is a technique that allows us to feel confident in our clustering solutions, given that it incorporates clustering results from various algorithms in a way that filters out errors. The final clustering provided by cluster aggregation may not always be the best solution, but we have consistently found it to be well above the accuracy of the average solution.

Clustering via dimension reduction provides us with hundreds of possible clustering results that can be fed into the aggregation process. As we have demonstrated, we can reduce the dimensions of a problem via NMF and SVD or both. The suggested dimension reduction algorithms use other clustering algorithms as part of their process, and often improve upon the accuracy of those algorithms used alone. With the various ways we can combine algorithms and the number of choices for r, the aggregation possibilities enumerate quickly.

Bibliography

[1] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*, chapter 8. Addison Wesley, 2005.

[2] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–91, October 1999.

[3] Amy Langville, Carl D. Meyer, Russell Albright, James Cox, and David Duling. Initializations, algorithms, and convergences for the nonnegative matrix factorization. Preprint.

[4] George H. Dunteman. *Principal Component Analysis*. Quantitative Applications in the Social Sciences. Sage Publications, May 1989.

[5] Daniel Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, December 1998.

[6] S Deerwester, ST Dumais, GW Furnas, TK Landauer, and R Harshman. Indexing by latent semantic analysis. *Journal for the American Society for Information Science*, 41(6):391–407, September 1990.

[7] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[8] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, March 1957.

[9] Mark Sinka and David Corne. A large benchmark dataset for web document clustering. *Soft Computing Systems: Design, Management and Applications*, 87:881–890, 2002.

[10] D Ziempekis and E Gallopoulos. Tmg: A matlab toolbox for generating term-document matrices from text collections. In J Kogan ad C Nicholas and M Teboulle,

editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 187–210. Springer, 2006.

[11] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[12] G Leitmann. On one approach to the control of uncertain system. *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, 50(5):373–380, 1993.