ABSTRACT

ABBEY, RALPH WALTER. Stochastic Clustering: Visualization and Application. (Under the direction of Dr. Carl D. Meyer.)

Data clustering is an important task in the field of data mining. In many cases clustering is used as an exploratory tool to better understand data. However, one difficulty in clustering is determining the quality of a given clustering result. While many clustering algorithms exist, we focus on a recently developed algorithm: stochastic clustering. We show how the results of the stochastic clustering can be visualized, and how this visualization indicates the quality of the clustering result.

One key step in stochastic clustering is to convert a nearly uncoupled similarity matrix into a doubly stochastic matrix. We propose an alternate iterative method to the commonly used Sinkhorn-Knopp algorithm, and provide error bounds in the limit. Additionally we develop stricter bounds than previously determined, that show the conversion to doubly stochastic form neither creates nor destroys the nearly uncoupled property.

We perform clustering experiments on a wide variety of data sets from different disciplines and of different sizes, and we compare the stochastic clustering to the popular spectral clustering method on these data sets. We consider multiple similarity matrices and conclude that the consensus similarity matrix is the most suitable for use in stochastic clustering.

Stochastic Clustering: Visualization and Application

by
Ralph Walter Abbey

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Mathematics

Raleigh, North Carolina

2014

APPROVED BY:

Dr. Ilse Ipsen	Dr. Rada Chirkova
Dr. Ernie Stitzinger	Dr. Carl D. Meyer Chair of Advisory Committee

DEDICATION

To my wife, who told me that not finishing is false advertising.

BIOGRAPHY

The author was born in a suburb of Cleveland Ohio on in 1985. Not a fan of the cold weather, the author convinced his family to move to Cary North Carolina soon after. The author went to elementary, middle, and high school in Raleigh, and first enrolled in classes at North Carolina State University as a junior in high school taking calculus 3 and differential equations. The author continued his college career at NC State with undergraduate degrees in mathematics and physics. A future wife, family nearby, a great advisor, and no desire to move (especially to a colder climate) all contributed to the author staying at NC State for graduate school. In a huge break from tradition, the author is not continuing on at NC State after his PhD program, but has accepted a full time position at SAS. The author will be trading an office in SAS Hall at NC State for an office on SAS campus in Cary NC, so hopefully the change will not be too difficult to handle.

ACKNOWLEDGEMENTS

I'd like to acknowledge my wife, my family, and my friends for their support, especially Zack Kenz and Miriam Diller during the thesis editing process. I'd like to thank the members on my committee for making time in their busy schedules for me. I'd like to thank Dr. Meyer for having me as a student and his advising and mentoring effort. I'm especially grateful to Dr. Meyer for his REU back in 2007 in which he introduced me to world of data mining.

I'd like to thank my academic siblings: Anjela Govan, Chuck Wessell, and Shaina Race, all former students of Dr. Meyer, and Hansi Jiang. Anjela and Chuck are the cool older academic sister and brother that I look up too, while Shaina is the academic sister who teases me, and who I can tease back.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Data Clustering

Data clustering, or just clustering, is the task of segmenting a set of observations into groups, such that similar observation are in the same groups, while dissimilar observations are in different groups. The set of observations we call the **data set**, and the groups we call **clusters**. The task of clustering can be a goal in itself, or part of a larger data mining and knowledge discovery process, where clustering is used for summarization, compression, or in finding nearest neighbors ([79] pg. 489).

The SEMMA model of data mining applications, which stands for Sample, Explore, Modify, Model, and Assess, places clustering in the Explore stage [6]. This is because clustering is inherently an exploratory task, in which a user seeks to gain new insight into the data.

Clustering is not just a task of academic curiosity, but also of interest to business and industry, as illustrated in commercial products such as SAS®Enterprise Miner™[70] and Radoop [65] which include tools for clustering. Clustering is useful in many computational fields such as pattern recognition, information retrieval, machine learning, classification, and bio-informatics.

Due to the shared interest in both academic and business communities, as well as the wide variety of fields in which it is used, clustering is a broad topic with many algorithms, both specialized and generic. One reason for the large quantity of algorithms is summarized nicely in the following statement.

Theorem 1.1.1. *([41] pg. xiv)*

There does not exist a best method, that is, one which is superior to all other methods, for

solving all problems in a given class of problems.

Additionally, as clustering is fundamentally an exploratory task, different methods can provide different results, all of which could be meaningful. While there may be many good clustering results, and many good clustering methods, we should be careful to not fall into the trap of thinking that all results and all methods are equally valid. While there may be multiple 'correct' answers in clustering, there are still wrong answers: grouping observations that are dissimilar together.

1.1.1 Motivating Example

Let us consider an industry application of clustering as a motivating example for our goals in this thesis. We consider a data analyst working for a major credit card company who wishes to create a model to predict which customers are most likely to churn; that is, to cancel their card. By creating a model that accurately predicts which customers may churn, the analyst enables the credit card company to offer special promotions to keep customers from leaving.

The data analyst uses clustering as an initial step to segment the customers into groups. She then creates a different classification model for each group separately, thereby improving the overall classification (application paraphrased from [20]). Ultimately the analyst helps the credit card company retain customers and increase profit.

Clustering is an important part of the predictive modeling process. A good clustering algorithm will group similar customers together, which can improve the predictive methods that the data analyst uses. The data analyst in our scenario wants several things from a clustering algorithm. Primarily, she wants the algorithm to work well at the clustering task: segmenting the customers into groups based on of similarities. She also knows that one specific clustering algorithm may not perform well in all cases. Thus her second desire is for a way to validate that the clustering algorithm performed as desired.

Most of the time, validation in this scenario comes at the end of the entire modeling process in the form of summary statistics, by means of lift charts, or through some measure, such as root mean square error (RMSE). However if the data analyst could validate the quality of the clustering, then she would be better able to determine where to focus her model creation efforts. In this thesis we will use data visualization for the purpose of validation of clustering. We focus on visualization because visual representation of results is much more understandable than strictly reporting numbers.

1.2 Data Visualization

The exploratory task of clustering is also highly tied to data visualization. Data visualization is the act of presenting representations of the data that a user can see. Visualization supports the human ability to look for patterns and relationships in the data ([26] pg. 21). It naturally occurs during the Explore phase of the SEMMA model, just as clustering does. Because there is no single clustering algorithm that we can turn to in all situations, having good cluster visualization methods can help indicate to a user whether a given clustering method is useful for a particular application.

Data visualization is used in many fields of science, and it includes many techniques that are specialized for a given purpose or data set (e.g. [30], and as can be seen in places such as [34] [60]). In this thesis we focus specifically on visualization of points in $\mathbb{R}^n$ without regard to a specific field or underlying structure. From our motivating example, we specifically want to investigate how visualization can aid in cluster validation.

Many generic methods have been proposed to visualize data, especially high dimensional data, including by using projections [52] [5] [31] [32] [3] [80], plotting parallel coordinates [88], using interactive methods [14], and using many other methods on which entire books have been written [26].

Clustering can aid in the visualization process as well, due to the fact that the act of clustering collects similar observations together, based off of patterns in the data. The data can be reorganized according to the determined clustering, thus allowing a more intuitive presentation of the data. In addition to the plethora of visualization methods for data, many methods have been proposed that use clustering in the visualization process [19] [2] [15] [83] [66]. Finally, some visualization methods, such as those in the Gephi software package [8], are designed for specific structures of data.

1.3 Similarity Graphs and Matrices

In many traditional clustering formulations a dataset of n observations and m numerical attributes is represented as an $m \times n$ matrix, M. An example of this formulation can be found in organizing textual documents, in which the attributes are the words and thus M_{ij} would be the number of times that word i appears in document j. Many of the aforementioned visualization techniques also use this representation in generating the visual representations of the data. An alternative way to represent the data is through the creation of similarity matrices.

Representing the data in terms of a similarity graph has potential benefits, as graph theoretical results now apply to the clustering problem. The clustering problem for the original data is equivalent to the graph partitioning problem, in which we want the edges between partitions

to have low weight.

Given a set of data points $x_i \in \mathbb{R}^n$ and a measure of similarity, a similarity graph is defined where each vertex v_i represents the data point x_i, and the edge between v_i and v_j has weight S_{ij}. If $S_{ij} = 0$ then there is no edge between v_i and v_j.

A similarity matrix S is a symmetric matrix in which S_{ij} is the similarity between x_i and x_j. This matrix is the adjacency matrix of the similarity graph. We will continue our discussions in terms of the similarity matrix, though all of these ideas can be traced back to the graphs that the similarity matrix describes.

We introduce a few measures of similarity here which will be used in later portions of the thesis.

- Gaussian kernel [42] similarity - The Gaussian kernel similarity matrix is used in spectral clustering [84]. This similarity matrix includes a user determined parameter, $\sigma > 0$ which is used to control the spread of the similarity values.

$$S_{ij} = e^{-\frac{\|x_i - x_j\|_2^2}{\sigma_{ij}}}$$

- Cosine similarity - The cosine similarity is often used with word document sets ([79] pg. 75).

$$S_{ij} = \frac{x_i^T x_j}{\|x_i\|_2 \|x_j\|_2}$$

- Shared K-nearest neighbor similarity [37] - The shared k-nearest neighbor similarity defines the similarity between two observations by the number of 'neighbors' they share. For each observation and a given distance metric, first construct the list of the K nearest neighbors. S_{ij} is the size of the intersection of the nearest neighbor lists for observation i and observation j. In [37] nearest refers to Euclidean distance, but this concept can be generalized to any distance measure.

- Consensus similarity - The consensus similarity matrix is different from the previous three similarity matrices in that it requires the observations to have previously been clustered by multiple algorithms, or multiple times by an algorithm with random initializations. The consensus similarity, as the name implies, tries creates a similarity in which the goal is to find a consensus between the previously-run clustering methods.

  S_{ij} is the number of times observation i and observation j are clustered together in multiple clusterings of the data set. Let $A^{(k)}$ be the adjacency matrix of the kth clustering of a data set. We define the elements

$$a_{ij}^{(k)} = \begin{cases} 1 & : \text{if observations } i \text{ and } j \text{ were clustered together} \\ 0 & : \text{if observation } i \text{ and } j \text{ were not clustered together} \end{cases}$$

Definition 1.3.1. *([86] pg. 5)*

If $\{A^{(1)}, A^{(2)}, \ldots, A^{(r)}\}$ is a collection of adjacency matrices created from clusterings of the same data set, then the sum of these matrices

$$S = A^{(1)} + A^{(2)} + \cdots + A^{(r)}$$

*is called the **consensus similarity matrix** or the consensus matrix.*

From our earlier discussion of clustering algorithms we know that many different clustering results can be obtained from a given data set. It is a natural desire, then, to want to find some agreement between different clustering results. The consensus similarity matrix is of particular interest due the fact that partitioning it acts as meta-clustering: we use previously determined clustering results to generate a graph we then seek to partition.

One recently developed method using the consensus similarity matrix is the stochastic consensus clustering algorithm. It is this method that we consider in this thesis.

1.4 Stochastic Consensus Clustering

The stochastic consensus clustering algorithm is shown below:

Algorithm 1.4.1. *Stochastic Consensus Clustering Algorithm [86]*

Input: *N observations $x_i \in \mathbb{R}^m$.*

1. *Create the consensus similarity matrix S using a clustering consensus of the user's choice*
2. *Use matrix balancing to convert S into a doubly stochastic symmetric matrix P.*
3. *Compute eigenvalues of P. Sort the eigenvalues of P in descending order and find the largest difference between successive eigenvalues. Let k be the number of eigenvalues before the largest difference.*
4. *Create a random s_0^T.*

5. *Track the evolution* $s_t^T = s_{t-1}^T P$. *After each iteration, sort the elements of* s_t^T *and then separate the elements into* k *clusters by dividing the sorted list at the* $k-1$ *largest gaps. When this clustering remains the same for a user-defined number of iterations, the final clusters are determined.*

Output: *Clusters* $C_1, \ldots, C_k$.

The stochastic consensus clustering algorithm (Wessell [86]) is a recently created algorithm that uses the consensus similarity matrix to perform data clustering. One of the original motivations is that stochastic consensus clustering acts as a way to create consensus among multiple clusterings, thus boosting user confidence in the final confidence of the result - a key factor in our motivating example. The second motivation is that the stochastic consensus clustering algorithm provides the user with k, the number of clusters, as opposed to requiring k as an input.

Other methods for determining k include the statistical technique called the *gap statistic* [82]; spectral clustering [84], which is another graph-based method; methods that look at the percentage of variance explained by different numbers of clusters [38] [81]; using cross-validation over multiple choices for k [78]; and optimizing a given metric over multiple choices of k [46].

1.4.1 Organization and Goals

All proofs in this thesis are original; non-original theorems are cited without proof. We will often use the phrase 'stochastic clustering algorithm' or 'stochastic consensus clustering algorithm.' The latter phrase refers specifically to the formulation in Algorithm 1.4.1, whereas the former is a more general case in which any similarity matrix is used (not just the consensus similarity - see step 1). This thesis investigates both the general and the specific cases.

In our motivating example the data analyst wants two main things: for her clustering method of choice to work well, and to be able to verify that it works well. The traditional method for showing a clustering method works well for a given application is to use data with known clusters and to show that the clustering obtains the known result.

We have three main goals in this thesis.

We wish to show that the stochastic clustering algorithm can generally perform well, which is mainly an experimental endeavor. We will do this by comparing the stochastic clustering algorithm to spectral clustering, a popular similarity matrix-based clustering method. We compare these two methods on several data sets that have a wide range in number of observations as well as a wide range in application areas.

We also wish to improve the original stochastic clustering algorithm in terms of speed and quality. In the future research section of [86], Wessell provides several areas of research for

which the stochastic clustering algorithm can be improved. We explore three of these questions: 1) Is there another similarity measure for which stochastic clustering will work? 2) Can the Sinkhorn-Knopp algorithm be replaced by one instance of row and column scaling? 3) Can we improve the theoretical bounds relating consensus matrix, S, to its doubly stochastic form, P? We address 1) experimentally and 2) by proposing a faster balancing algorithm to cut the potentially long convergence times for the Sinkhorn-Knopp.

Finally, we wish to provide a way for a user to validate the clustering output. We propose a method for visualizing the output of the stochastic clustering algorithm. We claim that this visualization allows a user to visually validate how the clustering is performing without having to decipher tables of numbers, or to inspect specific clustering results.

We organize this thesis as follows. In Chapter 2 of this thesis we focus on the theoretical background for the stochastic clustering algorithm. This theoretical background is necessary for our goals of improving the algorithm, as well as visualization. In Chapter 3 we show improvements for the bounds relating the consensus similarity matrix, S, to the doubly stochastic matrix, P. In addition, we provide a new algorithm that we argue will convert S to P faster than currently used methods, when appropriate prerequisites are met.

In Chapter 4 we introduce a method of visualizing the data and clustering from the stochastic consensus clustering method. We argue that through visualization a user can both validate the quality of clustering as well as improve upon the standard results of the stochastic clustering algorithm. We also discuss the initialization of the Markov chain related to step 4 of the stochastic clustering algorithm. We expand previous work on the theoretical issues that can cause poor initializations. Through the use of visualization we argue that it is possible to understand when a poor initialization has been encountered.

We discuss experimental setup and results in Chapters 5 and 6 respectively. In Chapter 6 we show experimentally that stochastic stochastic consensus clustering performs well in comparison to spectral clustering on several different datasets. We show that the consensus similarity matrix is the appropriate similarity matrix on which to use the stochastic clustering. We also show how visualization can provide valuable insight for the clustering process and validation.

Finally, we end this thesis with our concluding remarks in Chapter 7.

CHAPTER 2

Stochastic Consensus Clustering Background

The stochastic clustering algorithm investigated in this thesis is based upon the variable aggregation work of Herbert Simon and Albert Ando [74]. In Section 2.1 we define notation and terminology that are used in this thesis, as well as provide relevant theorems for the introduced terminology. In Section 2.1.4 we give a brief overview of the Simon-Ando theory, and we explain how we use its results in the stochastic clustering algorithm. We finish this chapter by showing that we can generate an appropriate matrix for the Simon-Ando theory from the similarity matrices discussed in Chapter 1.

2.1 Notation and Terminology

We start with general notation, after which we break notation down into the topic areas of eigenvalue notation and notation which deals with matrix structure.

Definition 2.1.1. *(Meyer [54], pg. 663)*

If $A \in \mathbb{R}^{m\times n}$ such that $a_{ij} > 0$ for all elements of A, then we say that A is a ***positive matrix****, which we denote by $A > 0$.*

Definition 2.1.2. *(Meyer [54], pg. 670)*

If $A \in \mathbb{R}^{m\times n}$ such that $a_{ij} \geq 0$ for all elements of A, then we say that A is a ***nonnegative matrix****, which we denote by $A \geq 0$.*

Sometimes we need to specify that a nonnegative matrix has at least one element $a_{ij} \neq 0$. We denote this by $A \neq 0$.

2.1.1 Eigenvalue Notation

Eigenvalues are used in this thesis during the discussion of convergence for some of the algorithms later presented. To avoid confusion as to which matrix to which an eigenvalue belongs, we use $\lambda_i(A)$ to denote that λ_i is an eigenvalue of A.

It is useful to be able to order the eigenvalues of a matrix. For any $n \times n$ matrix A we can order the eigenvalues by the magnitude of the modulus such that $|\lambda_1(A)| \geq |\lambda_2(A)| \geq \cdots \geq |\lambda_n(A)|$.

If we know that A has all real eigenvalues then we can also order by the eigenvalues such that $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$.

We use both orderings in this thesis, and as such the presence of the absolute value symbol, or lack thereof, will indicate which ordering is being used.

Definition 2.1.3. *(Meyer [54], pg. 497)*

*The **spectral radius** of a $n \times n$ matrix A, $\rho(A)$, is the maximum of the modulus of all eigenvalues of A. That is $\rho(A) = \max_i |\lambda_i(A)|$.*

2.1.2 Non-Zero Structure of a Matrix

We now define the terms **fully indecomposable**, **total support**, **irreducible**, and **primitive**, all of which deal with the structure of a matrix, or the locations of zeros and non-zeros in a matrix. We also provide some additional theorems that relate these concepts to each other. These terms are used in the theorems that relate to the Simon-Ando theory, and also to show how the stochastic clustering algorithm satisfies necessary assumptions.

Some concepts have multiple terms associated with them in the literature. For example **completely indecomposable** is used in [45], while in many other sources the term used is **fully indecomposable** [40] [13] [9]. The term **indecomposable** has also been used to have the same meaning as **irreducible** [21].

Definition 2.1.4. *(Meyer [54], pg. 671)*

*An $n \times n$ nonnegative matrix A is said to be **reducible** if there exists a permutation matrix Q such that*

$$Q^T A Q = \begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix}$$

*where X and Z are both square. If no such permutation exists then A is said to be **irreducible**.*

Definition 2.1.5. *(Meyer [54], pg. 674)*

*An irreducible matrix A is said to be a **primitive matrix** if $\lambda_1(A) = \rho(A)$, and $|\lambda_i(A)| < |\lambda_1(A)|$ for $i \neq 1$.*

Not all irreducible matrices are primitive, which we illustrate in Example 2.1.6.

Example 2.1.6. *(If A is an irreducible matrix, then A is not necessarily primitive.)*

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

It is clear that A is irreducible. However A has the eigenvalues of -1 and 1, and thus A is not primitive because $|\lambda_2(A)| \not< |\lambda_1(A)|$.

Theorem 2.1.7. *(Meyer [54] pg. 678)*

If A is an irreducible matrix, then A is primitive if and only if there exists an integer $k > 0$ *such that* $A^k > 0$.

Example 2.1.6 also illustrates Theorem 2.1.7, as $A^2 = I$. Therefore there is no k such that $A^k > 0$.

Theorem 2.1.8. *(Plemmons [9], pg. 34)*

If A is an irreducible matrix with a positive trace, then A is primitive.

It is worth noting that the converse of Theorem 2.1.8 is not necessarily true: if A is primitive, it is not generally required that A have a positive trace. We illustrate this in Example 2.1.9. In the case of 2×2 matrices it is true that A has a positive trace if A is primitive.

Example 2.1.9.

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad , \quad A^2 = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

In this case A does not have a positive trace, but A is primitive because $A^2 > 0$.

Several proofs that we discuss later, by Sinkhorn and Knopp, Brualdi, and also Ruiz, require a generalization of the concept of irreducible.

Definition 2.1.10. *(Minc [55] pg. 82)*

An $n \times n$ *nonnegative matrix A is said to be* ***fully indecomposable*** *if there are no permutation matrices* Q_1 *and* Q_2 *such that*

$$Q_1 A Q_2 = \begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix}$$

where X and Z are both square.

If a matrix is fully indecomposable then it is irreducible. However, the converse is not true, which we illustrate in Example 2.1.11.

Example 2.1.11. *(Irreducible does not imply fully indecomposable)*

Q_1 and Q_2 are the only 2×2 permutation matrices

$$Q_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad , \quad Q_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

If A is

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

then A is irreducible as neither $Q_1^T A Q_1$ nor $Q_2^T A Q_2$ are reducible. However, A is not fully indecomposable, as seen by computing $Q_2 A Q_1$.

$$Q_2 A Q_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

In order to more fully relate irreducible and fully indecomposable, we need the concept of total support.

Definition 2.1.12. *(Sinkhorn and Knopp [75] pg. 343)*

*Given an $n \times n$ real matrix A, and a permutation, σ, of the numbers $1, 2, \ldots, n$, the sequence of elements $a_{1\sigma(1)}, a_{2\sigma(2)}, \ldots, a_{n\sigma(n)}$ is the **diagonal** of A corresponding to σ. If σ is the identity permutation, then the diagonal is called the main diagonal.*

Definition 2.1.13. *(Sinkhorn and Knopp [75] pg. 343))*

*If A is an $n \times n$ nonnegative matrix, A is said to have **total support** if $A \neq 0$ and if every positive element of A lies on a positive diagonal (where $a_{1\sigma(1)}, a_{2\sigma(2)}, \ldots, a_{n\sigma(n)} > 0$ is one such diagonal).*

As much of the discussion in this thesis focuses on symmetric matrices, especially with a positive main diagonal, we relate these types of matrices to the concept of total support.

Lemma 2.1.14. *If A is a nonnegative symmetric matrix with a positive main diagonal, then A has total support.*

Proof. To prove this, we need to show that every positive element lies on a positive diagonal.

For $a_{ij} > 0$ we define σ_{ij} to be the permutation where $\sigma(i) = j$, $\sigma(j) = i$, and $\sigma(k) = k$ for all $k \neq i, j$. Because A has a positive main diagonal we know that if $k \neq i, j$ then $a_{k\sigma(k)} = a_{kk} > 0$. Since A is symmetric, if $a_{ij} > 0$, then $a_{i\sigma(i)} = a_{ij} = a_{ji} = a_{j\sigma(j)} > 0$. Thus a_{ij} lies on a positive diagonal. □

The following two lemmas provide some relationships between fully indecomposable and total support.

Lemma 2.1.15. *(Minc [55] pg. 83)*

Every positive entry of a fully indecomposable matrix lies on a positive diagonal.

Theorem 2.1.16. *(Csima and Datta [18] pg. 149)*

Let A be an irreducible symmetric matrix with total support. Then either A is fully indecomposable or else there exists a permutation matrix Q such that

$$QAQ^T = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix}$$

where B is fully indecomposable.

It is also noted that if A is a fully indecomposable matrix, then A is also primitive [12] [39] [9][55]. A proof can be found in [45].

Theorem 2.1.17. *(Lewin [45] pg. 756)*

If A is a fully indecomposable nonnegative matrix then $A^{n-1} > 0$.

We synthesize the independent theorems and definitions into one theorem that relates the concepts of irreducible, primitive, total support, and fully indecomposable. This is especially useful as the similarity matrices used in this thesis are symmetric.

Theorem 2.1.18. *A symmetric matrix A is fully indecomposable if and only if A is irreducible, primitive, and has total support.*

Proof. ($\Rightarrow$) Assume A is fully indecomposable. Then A is irreducible (Definition 2.1.10), A has total support (Lemma 2.1.15), and A is primitive (Theorem 2.1.17).

($\Leftarrow$) Assume A is irreducible, primitive, and has total support. Theorem 2.1.16 ensures that either A is fully indecomposable or A has the form

$$QAQ^T = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix},$$

where B is fully indecomposable and Q is a permutation matrix. Since A is primitive, any symmetric permutation of A must be primitive. Therefore A is fully indecomposable. □

We note that when A is not symmetric, the ($\Rightarrow$) direction is still true, but the ($\Leftarrow$) direction does not hold. We provide a counter example in Example 2.1.19.

Example 2.1.19. *(If A is an irreducible and primitive matrix with total support, then A is not necessarily fully indecomposable)*

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

We see that A has total support by considering the diagonal formed by (a_{31}, a_{22}, a_{13}) and the diagonal formed by (a_{31}, a_{23}, a_{12}). Each positive element of A lies on one of these two diagonals, and each of these diagonals is made up of strictly positive elements.

There exists no permutation matrix Q such that QAQ^T is reducible, and we also see that A is primitive because $A^3 > 0$. Thus we know that A is irreducible, primitive, and has total support. Given Q_1 and Q_2

$$Q_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \text{ and } Q_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

then

$$Q_1 A Q_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

The computation Q_1AQ_2 shows that A is not fully indecomposable, which finishes the counterexample.

2.1.3 Nearly Uncoupled Matrices

The Simon-Ando theory, and thus the stochastic clustering algorithm, relies on the concept of **nearly uncoupled** matrices.

Definition 2.1.20. *(Wessell [87] pg. 1216)*

*The $n \times n$ matrix A is called **uncoupled** if it can be symmetrically permuted to the form*

$$QAQ^T = \begin{bmatrix} A_{11}^\star & 0 & \dots & 0 \\ 0 & A_{22}^\star & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_{kk}^\star \end{bmatrix}$$

where the diagonal blocks $A_{ii}^\star$ are square with size $n_i \times n_i$, and $n_1 + n_2 + \cdots + n_k = n$.

A matrix that cannot be permuted to the form in Definition 2.1.20 is not uncoupled.

Theorem 2.1.21. *(Wessel [86] pg. 13)*

If A is symmetric matrix, then A is irreducible if and only if A is not uncoupled.

If A is a matrix that is 'very close' to an uncoupled matrix, then we say that A is **nearly uncoupled**. We give a definition for stochastic matrices in 2.1.5, and a more general definition in Chapter 3.

If the similarity matrix used as input to Algorithm 1.4.1 is nearly uncoupled then we can use the Simon-Ando theory to find clusters. We examine this theory in the next section.

2.1.4 Stochastic Matrices and the Simon-Ando Theory

Definition 2.1.22. *(Meyer [54], pg. 687) A **stochastic matrix** is a nonnegative matrix $P_{n\times n}$ in which each row sum is equal to 1.*

In some cases a matrix may be referred to as either 'row stochastic' or 'column stochastic' as an indication of the rows or columns summing to 1 respectively. A matrix whose row sums and column sums both equal 1 is said to be doubly stochastic.

We call a nonnegative vector π^T a probability row vector if $\|\pi^T\|_1 = 1$. If P is a stochastic matrix and π_0^T is a probability row vector then we can consider the evolution equation

$$\pi_t^T = \pi_{t-1}^T P, \tag{2.1}$$

where each π_t^T will also be a probability row vector.

Definition 2.1.23. *(Wessell [86], pg 14) The row vector π^T is a stationary distribution vector of the stochastic matrix P if it satisfies the equations*

$$\begin{aligned} \pi^T &= \pi^T P \\ \pi^T &\geq 0 \\ \pi^T e &= 1 \end{aligned}$$

where e is the vector of all ones.

If P is primitive then we are guaranteed that a stationary distribution vector π^T exists, and

$$\pi^T = \lim_{t\to\infty} \pi_0^T P$$

for any initial probability row vector π_0^T.

The Simon-Ando theory was initially developed in order to gain understanding of the long term behavior of the evolution equation (Equation 2.1) when P is nearly uncoupled. It was first used in modeling micro and macro economies, but has enjoyed a wide range of applications in other areas including urban design [69], the theory of multicellular evolution [73], computer queueing systems [17], and neuro-science [77]. Stochastic clustering seeks the opposite approach;

given a known long-term behavior of the evolution equation, the goal is to find clusters by observing the short-run and middle-run behavior.

The stochastic clustering method requires that the input similarity matrix be converted to a doubly stochastic matrix. We continue this chapter with the assumption that such a conversation is possible. We discuss the details of this conversion at the end of this chapter, and in Chapter 3 of this thesis.

Simon and Ando argued that if P is primitive and nearly uncoupled, then π_t^T will go through several distinct stages before approaching the limit as a stationary distribution vector. To formally define these stages, we next define the stochastic complement.

2.1.5 Stochastic Complementation

Definition 2.1.24. *(Meyer [53], pg. 2) Let P be an $n \times n$ irreducible stochastic matrix with the structure*

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1k} \\ P_{21} & P_{22} & \dots & P_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ P_{k1} & P_{k2} & \dots & P_{kk} \end{bmatrix}$$

in which all diagonal blocks are square. Then each P_{ii} has a stochastic complement in P defined by

$$C_{ii} = P_{ii} + P_{i\star}(I - P_i)^{-1}P_{\star i},$$

where P_i is the matrix obtained by deleting the ith row and ith column of blocks from P. $P_{i\star}$ is the ith row of blocks of P excluding P_{ii}, and $P_{\star i}$ is the ith column of blocks also excluding P_{ii}.

$$\begin{aligned} P &= \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix} \\ C_{22} &= P_{22} + P_{2\star}(I - P_2)^{-1}P_{\star 2} \\ C_{22} &= P_{22} + \begin{bmatrix} P_{21} & P_{23} & P_{24} \end{bmatrix} \begin{bmatrix} I - P_{11} & -P_{13} & -P_{14} \\ -P_{31} & I - P_{33} & -P_{34} \\ -P_{41} & -P_{43} & I - P_{44} \end{bmatrix}^{-1} \begin{bmatrix} P_{12} \\ P_{32} \\ P_{42} \end{bmatrix} \end{aligned}$$

Figure 2.1: An example of the computation for a stochastic complement (Wessell [86], pg. 16). The equation for the stochastic complement C_{22} when P is a matrix with four diagonal blocks.

Theorem 2.1.25. *(Meyer [53], pg. 22) For an $n \times n$ irreducible stochastic matrix of the form*

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1k} \\ P_{21} & P_{22} & \dots & P_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ P_{k1} & P_{k2} & \dots & P_{kk} \end{bmatrix},$$

let C be the uncoupled stochastic matrix

$$C = \begin{bmatrix} C_{11} & 0 & \dots & 0 \\ 0 & C_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C_{kk} \end{bmatrix},$$

and assume that each stochastic complement C_{ii} is primitive. Let the eigenvalues of C be ordered

$$\lambda_1 = \lambda_2 = \cdots = \lambda_k = 1 > |\lambda_{k+1}| \geq \cdots \geq |\lambda_n|,$$

and assume that C is similar to a diagonal matrix

$$Z^{-1}SZ = \begin{bmatrix} I_{k\times k} & 0 \\ 0 & D \end{bmatrix}.$$

If C^∞ denotes the limit

$$C^\infty = \lim_{t\to\infty} C^t = C = \begin{bmatrix} ec_1 & 0 & \dots & 0 \\ 0 & ec_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & ec_k \end{bmatrix},$$

where c_i is the stationary distribution vector for C_{ii} then

$$\|P^t - C^\infty\|_\infty \leq t2 \max_i \|P_{i\star}\|_\infty + \kappa|\lambda_{k+1}|^t$$

where

$$\kappa = \|Z\|_\infty \|Z^{-1}\|_\infty.$$

In addition, the difference between π_t^T and c^T where

$$c^T = \lim_{t\to\infty} \pi_0^T C^t$$

satisfies

$$\|\pi_t^T - c^T\|_1 \leq t2 \max_i \|P_{i\star}\|_\infty + \kappa|\lambda_{k+1}|^t.$$

The quantity $\max_i \|P_{i\star}\|_\infty$, given in the equation $2\max_i \|P_{i\star}\|_\infty = \|P - C\|_\infty$, measures of how far P is from the matrix C (Meyer [53] pg. 18). As C is uncoupled, we can think of $\max_i \|P_{i\star}\|_\infty$ as the degree of uncoupling for stochastic matrices. We propose a general definition for the degree of uncoupling in Chapter 3. An important corollary to Theorem 2.1.25 establishes bounds on t.

Corollary 2.1.26. *(Meyer [53], pg. 25) If, for any $\epsilon > 0$, t lies in the interval defined by*

$$\frac{\ln(\epsilon/2\kappa)}{\ln(|\lambda_{k+1}|)} < t < \frac{\epsilon}{4\max_i \|P_{i\star}\|_\infty} \tag{2.2}$$

then

$$\|P^t - C^\infty\|_\infty < \epsilon$$

and for every π_0^T,

$$\|\pi_t^T - c^T\|_1 < \epsilon.$$

The theory of stochastic complementation is very nuanced and can be found in full in [53]. In addition to a full exposition, explanation is provided as to why the primitivity assumption for P and C_{ii}, as well as the requirement that C is diagonalizable (found in the statement of Theorem 2.1.25) are not needed.

The theory for stochastic clustering relies on the existence of the interval given in Corollary 2.1.26. In a general setting we will not have information on many of the variables, such as λ_{k+1} and κ. In Chapter 3 we discuss $\max_i \|P_{i\star}\|_\infty$, and ensuring that this quantity is small. By requiring that $\max_i \|P_{i\star}\|_\infty$ is small, we will guarantee the interval for t to exists.

A consequence of Theorem 2.1.25 is that while t is in the bounds provided by Corollary 2.1.26, the probability vector π_t^T is given by

$$\pi_t^T \approx \left(\begin{array}{c|c|c|c} \alpha_1 c_1^T & \alpha_2 c_2^T & \ldots & \alpha_k c_k^T \end{array} \right).$$

We refer to this as the **short-run dynamics** of the evolution equation. Wessell showed ([86] pg. 22) that if an additional assumption that P is doubly stochastic is made, then

$$\begin{aligned} \pi_t^T &\approx \left(\begin{array}{c|c|c|c} \alpha_1 c_1^T & \alpha_2 c_2^T & \ldots & \alpha_k c_k^T \end{array} \right) \\ &= \left(\begin{array}{c|c|c|c} \frac{\alpha_1}{n_1}, \ \ldots, \ \frac{\alpha_1}{n_1} & \frac{\alpha_2}{n_2}, \ \ldots, \ \frac{\alpha_2}{n_2} & \ldots & \frac{\alpha_k}{n_k}, \ \ldots, \ \frac{\alpha_k}{n_k} \end{array} \right), \end{aligned}$$

where α_i depends on the initial probability vector π_0^T. Each c_i^T, the stationary distribution of the Cii stochastic complement, is a uniform probability vector of length n_i.

Theorem 2.1.27. *(Meyer [53] pg. 28) For an $n \times n$ irreducible stochastic matrix*

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1k} \\ P_{21} & P_{22} & \dots & P_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ P_{k1} & P_{k2} & \dots & P_{kk} \end{bmatrix}$$

with stochastic complements C_{ii} which are each primitive, let c_i^T be the stationary distribution vector for C_{ii} and let $\pi_t^T = \pi_0 P^t$ where π_0^T is an arbitrary initial distribution. If $\epsilon > 0$ is a number such that there exists a t within the bounds provided by Corollary 2.1.26, then for each integer larger than t, there exist scalars $\beta_i(t)$ such that

$$v_t^T = \left(\begin{array}{c|c|c|c} \beta_1(t)c_1^T & \beta_2(t)c_2^T & \dots & \beta_k(t)c_k^T \end{array} \right)$$

satisfies the inequality

$$\|\pi_t^T - v_t^T\|_1 < \epsilon.$$

We call this the **middle-run dynamics** for the evolution equation. For doubly stochastic matrices, Wessell showed ([86] pg. 22) that

$$\begin{aligned} \pi_t^T &\approx \left(\begin{array}{c|c|c|c} \beta_1(t)c_1^T & \beta_2(t)c_2^T & \dots & \beta_k(t)c_k^T \end{array} \right) \\ &= \left(\begin{array}{c|c|c|c} \frac{\beta_1(t)}{n_1}, \ \dots, \ \frac{\beta_1(t)}{n_1} & \frac{\beta_2(t)}{n_2}, \ \dots, \ \frac{\beta_2(t)}{n_2} & \dots & \frac{\beta_k(t)}{n_k}, \ \dots, \ \frac{\beta_k(t)}{n_k} \end{array} \right). \end{aligned}$$

If we have converted an input similarity matrix to be a doubly stochastic, then, assuming that it is nearly uncoupled, we can find the clusters by observing the iterates π_t^T in the short-run and middle-run dynamics ($t > \frac{\ln(\epsilon/2\kappa}{ln(|\lambda_{k+1}|)}$). While we do not know *a priori* what the α_i and $\beta_i(t)$ are, we know that there are groups of approximately equal probability values in π_t^T. If the ith and jth component of π_t^T, $\pi_t^T(i)$ and $\pi_t^T(j)$ respectively, are approximately equal then we say that observations i and j are in the same cluster. We create k clusters by dividing the sorted list of probability values in π_t^T at the $k-1$ largest (absolute, not relative) gaps.

An important consideration is if there exists a t within the bounds provided by Corollary 2.1.26. In addition, we want π_t^T to converge slowly to the stationary distribution so that we are able to observe the short-run and middle-run dynamics. Because $\pi_t^T = \pi_{t-1}^T P$ is a power method type iteration [85], we want the second largest eigenvalue of P to be close to 1 to ensure slow convergence. We examine these topics further in Chapter 3.

2.2 The Existence of a Doubly Stochastic Matrix, P

The stochastic consensus clustering algorithm stipulates that given a consensus matrix S we can construct a doubly stochastic matrix P to make use of the Simon-Ando theory. The following theorem states that if a general matrix A has the necessary properties, then A can be scaled to be doubly stochastic.

Theorem 2.2.1. *(Brualdi, et. al. [13] pg. 43)*

Let $A \geq 0$ be an $n \times n$ fully indecomposable matrix. Then there exist diagonal matrices D_1 and D_2 with positive diagonals such that D_1AD_2 is doubly stochastic. Moreover D_1 and D_2 are uniquely determined up to scalar multiples.

Unique up to a scalar multiple means that if $D_1AD_2 = P$ and $D_3AD_4 = P$, then $D_1 = \alpha D_3$ and $D_2 = \beta D_4$, where $\alpha\beta = 1$. It is also useful to keep in mind that P is unique if it exists (Sinkhorn and Knopp [75]). In the case of a symmetric matrix we have a unique scaling.

Lemma 2.2.2. *(Cisma and Datta [18] pg. 150)*

Let A be a fully indecomposable symmetric matrix. Then there exists a diagonal matrix D such that DAD is doubly stochastic.

The similarity consensus matrix S is nonnegative, symmetric, and has a positive main diagonal by construction. By Lemma 2.1.14, S has total support, and if S is also not uncoupled, thus irreducible, then it is fully indecomposable by Theorem 2.1.18. In this case we know that there exist unique diagonal matrix D and doubly stochastic matrix P such that $P = DSD$.

The Gaussian kernel, shared k-nearest neighbor, and cosine similarity matrices are also nonnegative, symmetric, and have positive main diagonal by construction. Thus if these matrices are not uncoupled, then they too are suitable for use in the stochastic consensus clustering algorithm in addition to the consensus similarity matrix. In general we will assume that an input similarity matrix is not uncoupled, as an uncoupled similarity matrix can be clustered trivially.

In Chapter 3 we show that the doubly stochastic matrix P formed from a given similarity matrix S is nearly uncoupled if S is nearly uncoupled. This is important, as the Simon-Ando theory only applies to nearly uncoupled stochastic matrices. We also discuss specific algorithms to create P from S, with the goal of computational efficiency.

CHAPTER 3

Matrix Scaling

Throughout this chapter we will use P to mean the doubly stochastic matrix formed by $P = DAD$, where A is a symmetric matrix with suitable properties discussed in Section 2.2. We focus on two main points: ensuring that P has the structure required for clustering using the Simon-Ando theory, and the process by which we obtain P from A.

In order to ensure that P has the suitable structure discussed in Chapter 2, we introduce an uncoupling measure from Wessell [86]. In addition we introduce our own new measure that relates to the stochastic complement and the theorems presented in Chapter 2. We relate these two measures, and also provide necessary bounds relating our new measure on A to our new measure on P.

Several methods for obtaining P from A have previously been developed [75] [40] [29]. Of these methods, the Sinkhorn-Knopp algorithm is the most well known (by citation count), and was used in stochastic consensus clustering. We argue that the rate of convergence of the Sinkhorn-Knopp algorithm guarantees only slow convergence for stochastic clustering. Instead we propose another algorithm for finding a doubly stochastic matrix, and we show that this new algorithm has a better rate of convergence than the Sinkhorn-Knopp algorithm when applied to the stochastic consensus clustering algorithm.

3.1 Properties of a Doubly Stochastic Matrix, P

Our application of the Simon-Ando theory requires that we have a nearly uncoupled, and symmetric stochastic matrix. We therefore want to show that if A has these desired properties,

then they are inherited by P. We start with three lemmas covering inherited properties.

Lemma 3.1.1. *(Wessell [87], pg. 1221) If A is an $n \times n$ symmetric matrix with total support and $P = DAD$ is doubly stochastic, then P is symmetric.*

Lemma 3.1.2. *(Wessell [86], pg. 31) If A is an $n \times n$ irreducible matrix and $P = DAD$ is doubly stochastic, then P is irreducible.*

Lemma 3.1.3. *If A is primitive and $P = DAD$ is doubly stochastic, then P is primitive.*

Irreducible and primitive are required properties for Theorem 2.1.25, while symmetry is required for stochastic clustering. In addition to these properties, we need for P to be 'nearly uncoupled,' which we previously defined as $\max_i \|P_{i\star}\|_\infty$. This definition does not make sense for a general nonnegative square matrix A, as A can take any value, whereas P is required to be stochastic.

Definition 3.1.4. *Let A be an $n \times n$ nonnegative, symmetric, and irreducible matrix, such that*

$$A = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1k} \\ A_{21} & A_{22} & \dots & A_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k1} & A_{k2} & \dots & A_{kk} \end{bmatrix}.$$

*Let $A_i^\star$ be the ith row of blocks, $A_{i\star}$ be the ith row of blocks of A excluding A_{ii}, and e be the vector of all ones of an appropriate size. Then the **row uncoupling measure of** A is given by*

$$\mu(A) = \max_{i,j} \frac{(A_{i\star}e)_j}{(A_i^\star e)_j} \tag{3.1}$$

The quantity $\mu(A)$ is the maximum over all ratios of the off-block diagonal row sums divided by the entire row sum. That is, $\mu(A)$ can also be written as

$$\mu(A) = \max_{k, n_{i\star}} \frac{\sum_{j \in n_{i\star}} a_{kj}}{\sum_{j \in [1,n]} a_{kj}} \tag{3.2}$$

where $n_{i\star}$ corresponds to the set of all n not in the ith diagonal block.

It is worth noting that $\mu(A)$ will always be less than or equal to 1, where equality only holds if each element of the A_{ii} block is equal to 0.

In the case of stochastic matrices, the ratio $\max_{i,j} \frac{(P_{i\star}e)_j}{(P_i^\star e)_j} = \max_i \|P_{i\star}\|_\infty$, as each row sum of the stochastic matrix is equal to 1. Now that we have an uncoupling measure that is well defined for general matrices, and that also fits the theory of stochastic complementation, let

us consider the relationship between the row uncoupling measure of A and row uncoupling measure of P when $P = DAD$.

Theorem 3.1.5. *If A is an $n \times n$ nonnegative, symmetric, and irreducible matrix such that there exists a $P = DAD$, then*

$$\frac{\min_m d_{mm}}{\max_m d_{mm}}\mu(A) \leq \mu(P) \leq \frac{\max_m d_{mm}}{\min_m d_{mm}}\mu(A),$$

where d_{ii} is a diagonal element of D.

Proof. Since $P = DAD$ we know that $p_{ij} = d_{ii}a_{ij}d_{jj}$. Let us consider Equation 3.2.

$$\mu(P) = \max_{k,n_{i\star}} \frac{\sum_{j\in n_{i\star}} p_{kj}}{\sum_{j\in[1,n]} p_{kj}} = \max_{k,n_{i\star}} \frac{\sum_{j\in n_{i\star}} d_{kk}d_{jj}a_{kj}}{\sum_{j\in[1,n]} d_{kk}d_{jj}a_{kj}}$$

As the summations are only over j we can factor out and cancel the d_{kk}.

$$\mu(P) = \max_{k,n_{i\star}} \frac{\sum_{j\in n_{i\star}} d_{jj}a_{kj}}{\sum_{j\in[1,n]} d_{jj}a_{kj}}$$

We now replace d_{jj} with either the $\max_m d_{mm}$ or $\min_m d_{mm}$ dependent on the direction of the inequality.

$$\frac{\min_m d_{mm}}{\max_m d_{mm}} \max_{k,n_{i\star}} \frac{\sum_{j\in n_{i\star}} a_{kj}}{\sum_{j\in[1,n]} a_{kj}} \leq \mu(P) \leq \frac{\max_m d_{mm}}{\min_m d_{mm}} \max_{k,n_{i\star}} \frac{\sum_{j\in n_{i\star}} a_{kj}}{\sum_{j\in[1,n]} a_{kj}}$$

Substituting in the definition of $\mu(A)$ finishes our proof. □

In the case of consensus similarity matrices generated by r clustering runs, we can additionally bound d_{mm} and the ratios used in Theorem 3.1.5.

Theorem 3.1.6. *If S is the $n \times n$ irreducible consensus matrix created from r clustering results and there exists a doubly stochastic matrix $P = DSD$, then*

$$\frac{\max_m d_{mm}}{\min_m d_{mm}} \leq (n-1).$$

Proof. Since S is a consensus matrix, we know that $s_{ii} = r$. We also know that $\sum_j d_{ii}d_{jj}s_{ij} = 1$ for all i because P is stochastic. Thus by considering only the diagonal element we see that

$$d_{ii}^2 \leq \frac{1}{r}.$$

If we replace d_{ij} with $\frac{1}{\sqrt{r}}$ we also see that

$$\begin{aligned} 1 &\leq d_{ii} \sum_j \frac{1}{\sqrt{r}} s_{ij} \\ \frac{\sqrt{r}}{\sum_j s_{ij}} &\leq d_{ii} \\ \frac{\sqrt{r}}{\|S\|_\infty} &\leq d_{ii}. \end{aligned}$$

Assuming that there were no trivial clustering results in which all observations were clustered together, then $\|S\|_\infty \leq (n-1)r$. We are left with the result

$$\frac{\sqrt{r}}{r(n-1)} \leq d_{ii}.$$

Finally as the bounds on d_{ii} hold for all i, we substitute $\frac{\sqrt{r}}{r(n-1)} \leq d_{ii}$ into $\frac{\max_m d_{mm}}{\min_m d_{mm}}$, and see that

$$\frac{\max_m d_{mm}}{\min_m d_{mm}} \leq \frac{\frac{\sqrt{r}}{r(n-1)}}{\frac{1}{\sqrt{r}}} = (n-1).$$

□

It is important to note that Theorems 3.1.5 and 3.1.6 relate $\mu(A)$ to $\mu(P)$. From the theory of stochastic complementation we need $\mu(P)$ to be small. Specifically, we see that in Corollary 2.1.26, a small $\mu(P)$ will increase the likelihood that there is a nonempty interval in which $\frac{\ln(\epsilon/2\kappa)}{\ln(|\lambda_{k+1}|)} < t < \frac{\epsilon}{4 \max_i \|P_{i\star}\|_\infty}$.

While Theorem 3.1.5 does not strictly guarantee that if S is uncoupled then P is uncoupled, it does let us know that if we wish $\mu(P) \leq \epsilon$ we need at most $\mu(S) \leq \frac{\epsilon}{n}$.

This concludes our discussion on which properties of A that P inherits. We have shown that the doubly stochastic matrix $P = DAD$ will have the necessary properties for stochastic clustering if we ensure that A is symmetric, irreducible, primitive, and nearly uncoupled.

3.1.1 The 'Uncoupling Measure' and the 'Row Uncoupling Measure'

A different measure for the uncoupling of a matrix was used in the original discussion of the stochastic clustering algorithm.

Definition 3.1.7. *(Wessell [87] pg. 1221)*

Let n_1 and n_2 be fixed positive integers such that $n_1 + n_2 = n$, and let A by an $n \times n$ irreducible matrix whose respective rows and columns have been symmetrically permuted to the

form

$$QAQ^T = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

where A_{11} *is* $n_1 \times n_1$ *and* A_{22} *is* $n_2 \times n_2$ *so that the ratio*

$$\sigma(A, n_1) = \frac{e^T A_{12} e + e^T A_{21} e}{e^T A e}$$

is minimized over all symmetric permutations of A*. The quantity* $\sigma(A, n_1)$ *is called the* ***uncoupling measure of*** A ***with respect to parameter*** n_1*.*

There are two important results based upon the definition of $\sigma(A, n_1)$, which relate the second largest eigenvalue of the doubly stochastic matrix P and the uncoupling measure defined in Definition 3.1.7.

Theorem 3.1.8. *(Wessell [87] pg. 1222)*

For a fixed integer $n > 0$*, consider an* $n \times n$ *irreducible, symmetric, and doubly stochastic matrix* P*. Given* $\epsilon > 0$*, there exists a* $\delta > 0$ *such that if* $\sigma(P, n_1) < \delta$ *then* $|\lambda_2(P) - 1| < \epsilon$*.*

Theorem 3.1.9. *(Wessell [87] pg. 1223)*

For a fixed integer $n > 0$*, consider an* $n \times n$ *irreducible, symmetric, and doubly stochastic matrix* P*. Given* $\epsilon > 0$*, there exists a* $\delta > 0$ *such that if* $|\lambda_2(P) - 1| < \delta$ *then* $\sigma(P, n_1) < \epsilon$*.*

While the relation between ϵ and δ in the above proofs are not presented, the result does tell us that $\sigma(P, n_1)$ is small if and only if the second largest eigenvalue is also small [87]. It is important to note that neither $\mu(P)$ and $\sigma(P, n_1)$, nor $\mu(A)$ and $\sigma(A, n_1)$ can be computed a priori due to the effect of permutations. These notions of uncoupling measures exist for theoretical proofs relating the nearly uncoupled nature of A and P. Theorems 3.1.8 and 3.1.9 provide an actual method for testing the nearly uncoupled nature of doubly stochastic matrices.

Because we need the results of Theorems 3.1.8 and 3.1.9, we must relate the uncoupling measure to the row uncoupling measure.

Theorem 3.1.10. *If* A *is an* $n \times n$ *nonnegative, symmetric, and irreducible matrix, such that*

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

where A_{11} *is of size* $n_1 \times n_1$*, then*

$$\sigma(A, n_1) \leq \mu(A).$$

Proof. We complete this proof in two steps:

1. Let α_1, α_2, β_1, β_2, and γ be positive numbers such that

$$0 < \frac{\alpha_1}{\beta_1} \leq \frac{\alpha_2}{\beta_2} = \gamma.$$

Then $\alpha_1 \leq \beta_1\gamma$ and $\alpha_2 = \beta_2\gamma$. Adding α_1 and α_2 we see that

$$\alpha_1 + \alpha_2 \leq (\beta_1 + \beta_2)\gamma,$$

which gives us

$$\frac{\alpha_1 + \alpha_2}{\beta_1 + \beta_2} \leq \frac{\alpha_2}{\beta_2}.$$

2. Let $\alpha_k = \sum_{j \in n_{i\star}} a_{kj}$ and $\beta_k = \sum_{j \in [1,n]} a_{kj}$. By Definition 3.1.7 we see that

$$\sigma(A, n_1) = \frac{\sum_k \alpha_k}{\sum_k \beta_k}.$$

Using part 1) gives us the result $\sigma(A, n_1) \leq \mu(A)$.

□

3.2 Algorithms for Obtaining a Doubly Stochastic Matrix, P

The Sinkhorn-Knopp algorithm is probably the best known method by citation count for finding a doubly stochastic matrix. However there exist a variety of other algorithms which, while different, will find the same doubly stochastic matrix (Parlett and Landis [61] & Ruiz [67]). An advantage of the Sinkhorn-Knopp algorithm is that extensive work has been done in order to determine the rate of convergence of the algorithm (Knight [39], Franklin and Lorenz [29], & Soules [76]).

We present the Sinkhorn-Knopp algorithm and some of the theorems and properties of the algorithm. We then explain why the Sinkhorn-Knopp algorithm is not the optimal algorithm with respect to rate of convergence for stochastic consensus clustering.

3.2.1 Sinkhorn-Knopp Algorithm

Algorithm 3.2.1. *Sinkhorn-Knopp Algorithm*

Input: *Nonnegative, square matrix A with total support. Let $A_0 = A$.*
Start at $k = 1$

1. *Divide each element of in the ith row of A_{k-1} by the ith element of matrix-vector product $A_{k-1}e$. Let this equal $A_{k-1/2}$.*

2. *Divide each element in the jth column of $A_{k-1/2}$ by the jth element of vector-matrix product $e^T A_{k-1/2}$. Let this equal A_k.*

3. *Repeat steps 1) and 2) until $\|A_k - A_{k-1}\| < \tau$, where the norm and the parameter τ are the user's choice.*

Output: A_k

The limit $\lim_{k\to\infty} A_k = P$, where P is a doubly stochastic matrix [75]. While this algorithm describes a sequence of matrices A_k, it can be reformulated into an algorithm in terms of a sequence of vectors. In order to do so, we first need the definition of the diagonal operator.

Definition 3.2.2. *The **diagonal operator** $D(x) : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ creates an $n \times n$ diagonal matrix in which $d_{ii} = x_i$.*

Considering the steps in Algorithm 3.2.1 we see that $A_{k-1/2} = A_{k-1}D(r_{k-1})^{-1}$, where $r_{k-1} = A_{k-1}e$. Likewise $A_k = D(c_{k-1})^{-1}A_{k-1/2}$, where $c = e^T A_{k-1/2}$. Because the sequence of matrices A_k converges to the doubly stochastic matrix P, we see that r_k and c_k should also converge to an r and c respectively.

The equations

$$c_{k+1} = D(A^T r_k)^{-1}e, \quad r_{k+1} = D(Ac_{k+1})^{-1}e \tag{3.3}$$

are equivalent to the Sinkhorn-Knopp algorithm when $r_0 = e$ [39], and $\lim_{k\to\infty} r_k = r$ and $\lim_{k\to\infty} c_k = c$.

When A is symmetric, then the Equations in 3.3 can be combined into a single expression

$$x_{k+1} = D(Ax_k)^{-1}e \tag{3.4}$$

with $x_0 = e$. However, $\lim_{k\to\infty} x_k$ does not converge. Instead, the sequence of x_k consists of two convergent sub-sequences, given by $x_{2k} = r_k$ and $x_{2k+1} = c_k$, where r_k and c_k are from equation 3.3.

As each iteration of 3.4 corresponds to either a row normalization or a column normalization in the original Sinkhorn-Knopp algorithm, the computation of equation 3.4 can by coded in MATLAB as x = 1./(A*x).

Lemma 3.2.3. *(Knight [39], pg. 265)*

Suppose that A is a symmetric nonnegative fully indecomposable matrix. Then there is a unique positive vector $x_\star$ such that $D(x_\star)AD(x_\star) = P$, where P is doubly stochastic.

In the case of the Sinkhorn-Knopp algorithm we note that $x_\star = \sqrt{D(r)D(c)}e$. While $D(r)$ and $D(c)$ are unique only up to a scalar multiple, $D(x_\star)$ is unique, which allows us to consider the rate at which the error decreases through the iterative process.

Theorem 3.2.4. *(Knight [39] pg. 267)*

Suppose that A is a symmetric nonnegative and fully indecomposable matrix, and that $x_\star$ is the unique positive vector such that $D(x_\star)AD(x_\star) = P$, where P is doubly stochastic. Let x_k be the sequence generated by equation 3.4 with $x_0 = e$. Then for all $\epsilon > 0$ there exists a $K_1 \in \mathbb{Z}$ such that if $k \geq K_1$ then $x_k = \alpha_k x_\star + d_k$, where $\|d_k\| < \epsilon$ and α_k is bounded. Furthermore, there exists $K_2 \in \mathbb{Z}$ such that if $k \geq K_2$

$$\|d_{k+2}\| \leq |\lambda_2(P)|^2 \|d_k\|$$

In theorem 3.2.4 $\lambda_2(P)$ is the second largest, in magnitude, eigenvalue of P. The error in each iteration of the equation 3.3, $\|d_{k+2}\|$, is bounded by $|\lambda_2(P)|$ and the error of the iteration twice previous, $\|d_k\|$. The reason for $\|d_k\|$ is due to the alternate normalizing of rows and columns and our comments relating equations 3.3.

For doubly stochastic matrices built on nearly uncoupled matrices we have some prior knowledge of $|\lambda_2(P)|$. In considering Theorems 3.1.8 and 3.1.9, we see that nearly uncoupled matrices have $|\lambda_2(P)|$ close to 1. This means that our theoretical bound can only guarantee a slow convergence for the Sinkhorn-Knopp algorithm (Algorithm 3.2.1). We will explore experimental results showing this in Chapter 6.

While slow convergence is a desired property of the evolution of the stochastic process for the stochastic consensus clustering algorithm, it is not desired during the steps to produce the doubly stochastic matrix P. It is possible that in a severe case, the Sinkhorn-Knopp algorithm will be theoretically convergent, but will take a very long time to converge.

3.3 Alternative to the Sinkhorn-Knopp

Because the Sinkhorn-Knopp algorithm may converge slowly, we wish to consider other methods for obtaining a doubly stochastic P. We know that given a symmetric nonnegative matrix A which is fully indecomposable, then there exists a positive vector $x_\star$ such that $D(x_\star)AD(x_\star) = P$ (lemma 3.2.3). We propose the following algorithm.

Algorithm 3.3.1. *Simultaneous p-Scaling*

Input: *An $n \times n$ symmetric matrix A with total support, $p \in (0,1)$, a stopping criterion τ, and a vector norm.*

Let $x_0 = e$, and start at $k = 1$

1. $x_k = D(Ax_{k-1})^{-p} D(x_{k-1})^{1-p} e$
2. *Repeat step 1) until $\|x_k - x_{k-1}\| < \tau$, where the norm and the parameter τ are the user's choice.*

Output: x_k

We need to show that algorithm 3.3.1 converges to $x_\star$ if we are to propose it as an alternative to the Sinkhorn-Knopp algorithm. We begin by showing that $x_\star$ is a fixed point of equation $x_k = D(Ax_{k-1})^{-p} D(x_{k-1})^{1-p} e$.

Lemma 3.3.2. *If A is a nonnegative symmetric and fully indecomposable matrix, and $x_\star$ is a positive vector such that $D(x_\star)AD(x_\star) = P$, a doubly stochastic matrix, then $x_\star$ is a fixed point for the function*

$$f(x) = D(Ax)^{-p} D(x)^{1-p} e$$

Proof. We first note that if $P = D(x_\star)AD(x_\star)$ and P is doubly stochastic, then $AD(x_\star)e = Ax_\star$ implies that $D(Ax_\star) = D(x_\star)^{-1}$. Substituting $x_\star$ into $f(x)$ gives us

$$\begin{aligned} f(x_\star) &= D(Ax_\star)^{-p} D(x_\star)^{1-p} e \\ &= D(x_\star)^{p} D(x_\star)^{1-p} e \\ &= D(x_\star) e \\ &= x_\star. \end{aligned}$$

□

We have shown that $f(x_\star) = x_\star$, and now we want to show that $f(x) = D(Ax)^{-p} D(x)^{1-p} e$ will converge. First we need a statement about differentiability for a general function.

Definition 3.3.3. *(Lang [43] pg. 463)*

Let U be an open subset in E and let $x \in U$. Let $f : U \to F$ be a map. We say that f is ***differentiable*** *at x if there exists a continuous linear map $\Lambda : E \to F$ and a map θ defined for sufficiently small $\|h\| \in F$ such that*

$$\lim_{\|h\| \to 0} \frac{\theta(h)}{\|h\|} = 0,$$

and such that

$$f(x+h) = f(x) + \Lambda(h) + \theta(h)$$

The function $x = D(Ax)^{-p}D(x)^{1-p}e$ is a function from $\mathbb{R}^n$ to $\mathbb{R}^n$, which allows us to make an extra statement about the nature of the linear map.

Theorem 3.3.4. *(Lang [43] pg. 467)*

Let U be an open set of $\mathbb{R}^n$ and let $f : U \to \mathbb{R}^m$ be a map which is differentiable at x. Then the linear map $f'(x)$ is represented by the matrix

$$J_f(x) = \frac{\delta f_i(x)}{\delta x_j}$$

called the Jacobian matrix of f at x.

Taking definition 3.3.3 and theorem 3.3.4 we now have

$$f(x+h) = f(x) + J_f(x)h + \theta(h).$$

as the Jacobian has taken the place of the linear map Λ. With this knowledge we now consider a general theorem on the convergence of a function from $\mathbb{R}^n$ to $\mathbb{R}^n$.

Theorem 3.3.5. *(Quarteroni [62] pg. 295)*

Suppose that $f : B \subset \mathbb{R}^n \to \mathbb{R}^n$ has a fixed point $x_\star$ in the interior of B and that f is continuously differentiable in a neighborhood of $x_\star$. Denote by J_f the Jacobian matrix of f and assume that $\rho(J_f(x_\star)) < 1$. Then there exists a neighborhood C of $x_\star$ such that $C \subset B$ and, for any $x_0 \in C$, the iterates defined by $x_{k+1} = f(x_k)$ all lie in B and converge to $x_\star$.

We say that f in theorem 3.3.5 is **locally convergent**, as it converges for any $x_0 \in C$. We want to know what the values of p are for which the iterative schema are locally convergent. We do this by looking at how the values of p affect $\rho(J_f(x_\star))$, though first we must find the Jacobian of f.

Lemma 3.3.6. *If A is a nonnegative, symmetric, and fully indecomposable matrix, and $x_\star$ is a positive vector such that $D(x_\star)AD(x_\star) = P$ where P is doubly stochastic, and*

$$f(x) = D(Ax)^{-p}D(x)^{1-p}e,$$

then

$$J_f(x_\star) = (1-p)I - pD(x_\star)PD(x_\star)^{-1}$$

where $J_f(x_\star)$ is the Jacobian of $f(x)$, evaluated at $x_\star$.

Proof. Looking at the components of $f(x)$ we see that

$$f_i(x) = \frac{(x_i)^{1-p}}{(A_i x)^p}$$

where A_i is the ith row of A. The Jacobian is defined by $J_{i,j}(x) = \frac{\delta f_i(x)}{\delta x_j}$, and so we see that:

$$\begin{aligned} J_{i,j}(x) &= -p(x_i)^{1-p}(A_i x)^{-(1+p)}A_{i,j}, \quad i \neq j \\ J_{i,i}(x) &= (1-p)(x_i)^{-p}(A_i x)^{-p} - p(x_i)^{1-p}(A_i x)^{-(1+p)}A_{i,j}, \\ J(x) &= (1-p)D(Ax)^{-p}D(x)^{-p} - pD(Ax)^{-(1+p)}D(x)^{(1-p)}A \end{aligned}$$

Substituting the fixed point $x_\star$ into the Jacobian of f we see

$$\begin{aligned} J_f(x_\star) &= (1-p)D(Ax_\star)^{-p}D(x_\star)^{-p} - pD(Ax_\star)^{-(1+p)}D(x_\star)^{(1-p)}A \\ &= (1-p)I - pD(x_\star)^2 A \\ &= (1-p)I - pD(x_\star)D(x_\star)AD(x_\star)D(x_\star)^{-1} \\ &= (1-p)I - pD(x_\star)PD(x_\star)^{-1} \end{aligned}$$

□

Now that we have calculated the Jacobian we can investigate the spectral radius as it changes with p.

Theorem 3.3.7. *If A is a nonnegative symmetric and fully indecomposable matrix, and $x_\star$ is a positive vector such that $D(x_\star)AD(x_\star) = P$ where P is doubly stochastic, and*

$$f(x) = D(Ax)^{-p}D(x)^{1-p}e,$$

then $\rho(J_f(x_\star)) < 1$ for $0 < p < 1$. Furthermore

$$\rho(J_f(x_\star)) = \begin{cases} |1 - p(1+\lambda_n(P))| & \text{if } p \leq \frac{2}{3+\lambda_n(P)} \\ |1-2p| & \text{if } p > \frac{2}{3+\lambda_n(P)} \end{cases} \tag{3.5}$$

where $\lambda_n(P) = \min_i \lambda_i(P)$.

Proof. The spectral radius of the Jacobian $\rho(J_f(x_\star))$ depends on the eigenvalues of $D(x_\star)PD(x_\star)^{-1}$, which are the same eigenvalues as P, as they are similar matrices.

Because A is nonnegative, symmetric, and fully indecomposable, then P is as well. By the Perron-Frobenius theorem we know that $\rho(P) = 1$, and because P is primitive, -1 cannot be

an eigenvalue of P. P is real and symmetric, which guarantees that all of the eigenvalues of P are real.

The previous statements together show that all the eigenvalues of $D(x_\star)PD(x_\star)^{-1}$ are in the interval $(-1, 1]$. Now that we know this information about the eigenvalues we consider the Jacobian matrix:

$$\rho(J_f(x_\star)) = \max_i |\lambda_i(J_f(x_\star))|.$$

We substitute the result of lemma 3.3.6 in for $J_f(x_\star)$ and simplify

$$\begin{aligned}
\rho(J_f(x_\star)) &= \max_i |\lambda_i((1-p)I - pD(x_\star)PD(x_\star)^{-1})| \\
&= \max_i |\lambda_i((1-p)I) - \lambda_i(pD(x_\star)PD(x_\star)^{-1})| \\
&= \max_i |1 - p(1 + \lambda_i(P))|
\end{aligned}$$

Since every eigenvalue of P is in the interval of $(-1, 1]$ it is easy to see that for values of $p \leq 0$ or values of $p \geq 1$ the spectral radius of the Jacobian is greater than or equal to one. With a few minor calculations we find that

$$\rho(J_f(x_\star)) = \begin{cases} |1 - p(1 + \lambda_n(P))| & \text{if } 0 < p \leq \frac{2}{3+\lambda_n(P)} \\ |1 - 2p| & \text{if } 1 > p > \frac{2}{3+\lambda_n(P)}. \end{cases} \tag{3.6}$$

□

It is interesting to note that in the case of $p = 0$ the function $f(x) = D(x)e$, which is just the identity function. In the case of $p = 1$ the function $f(x) = D(Ax)^{-1}e$, which is the Sinkhorn-Knopp algorithm.

The results from theorem 3.3.7 only guarantee that if some x_0 is chosen "close" to $x_\star$, then the fixed point iteration converges. This is not enough to prove the convergence of algorithm 3.3.1. In the next section we show that the simultaneous scaling method given in [67] is equivalent to the simultaneous p-scaling algorithm with $p = \frac{1}{2}$.

3.3.1 Simultaneous Scaling

We first present the simultaneous scaling algorithm, and theorems that pertain to the algorithm. We then show that if A is symmetric, this algorithm is equivalent to simultaneous p-scaling (algorithm 3.3.1) for $p = \frac{1}{2}$ and $x_0 = e$.

Algorithm 3.3.8. *Simultaneous Scaling (Ruiz [67])*

Input: *An $n \times n$ symmetric matrix A with total support, a stopping criterion ϵ, and a vector norm.*

Let $A_0 = A$, $D_0 = I$ and $E_0 = I$, and start with $k = 1$.

1. *Compute the following:*

$$D_R = D(A_k e)^{-\frac{1}{2}} \quad , \quad D_C = D(e^T A_k)^{-\frac{1}{2}}$$
$$D_{k+1} = D_k D_R$$
$$E_{k+1} = E_k D_C$$
$$A_{k+1} = D_k A E_k$$

2. *Repeat 1) until*

$$\|A_k e - e\| \leq \epsilon \quad \textit{and} \quad \|e^T A_k - e^T\| \leq \epsilon$$

for a norm and ϵ of the user's choice.

Output: *A_k*

Theorem 3.3.9. *(Ruiz [67])*

If A has total support, then for the sequence of diagonal matrices D_k and E_k for A, described in algorithm 3.3.8, then both $D = \lim_{k\to\infty} D_k$ and $E = \lim_{k\to\infty} E_k$ exist and $P = DAE$ is doubly stochastic.

The simultaneous scaling algorithm and accompanying theorem give us an alternative to the Sinkhorn-Knopp algorithm. As noted in theorem 2.2.1, if there is a doubly stochastic matrix such that $P = DAE$, then P is unique. Therefore we know that the simultaneous scaling algorithm and the Sinkhorn-Knopp algorithm have the same doubly stochastic matrix limit.

In the case where A is symmetric, we can see that $D_R = D_C$ and $D_k = E_k$ in algorithm 3.3.8.

Corollary 3.3.10. *Let A be nonnegative, symmetric, and fully indecomposable, and let x_k be computed via the simultaneous p-scaling algorithm (algorithm 3.3.1) with $p = \frac{1}{2}$. If P is a doubly stochastic matrix such that $D(x_\star)AD(x_\star)$, then*

$$\lim_{k\to\infty} x_k = x_\star.$$

Proof. We know that theorem 3.3.9 guarantees that algorithm 3.3.8 converges. We therefore just need to show that the fixed point iteration produces the same iterates as algorithm 3.3.8.

Since A is symmetric we know that $D_R = D_C$ and $D_k = E_k$. We use the diagonal operator and define $D_R = D(x)$ and $D_k = D(x_k)$. We then write the update for D_k as

$$D(x_{k+1}) = D(x_k)D(x).$$

We substitute in $D(x) = D(A_k e)^{-\frac{1}{2}}$, as well as $A_k = D(x_k)AD(x_k)$, to get

$$D(x_{k+1}) = D(x_k)D(D(x_k)AD(x_k)e)^{-\frac{1}{2}}.$$

Given two vectors a and b, the diagonal operator has the property that $D(D(a)b) = D(a)D(b)$.

$$\begin{aligned} D(x_{k+1}) &= D(x_k)D(x_k)^{-\frac{1}{2}}D(Ax_k)^{-\frac{1}{2}} \\ &= D(Ax_k)^{-\frac{1}{2}}D(x_k)^{\frac{1}{2}}. \end{aligned}$$

We see the initial condition in that $D_0 = I$ corresponds to $x_0 = e$ as $D(x_0) = D_0$. □

Recall that the Sinkhorn-Knopp algorithm can be coded in MATLAB as x = 1./(A*x). For the simultaneous scaling, the computation can be coded in MATLAB as x = (x./(A*x)).∧(-1/2).

By comparing these two implementations it is easy to see that each iteration (one row scaling or one column scaling) of Sinkhorn-Knopp requires $n^2 + n$ operations. Each iteration of the simultaneous scaling requires $n^2 + 2n$ operations.

While we have not shown that the simultaneous p-scaling algorithm is convergent for all $p \in (0, 1)$ we have noticed that it does converge in the examples we have seen in practice. In Chapter 6 we will show some of these results, though mathematically we can only be guaranteed global convergence in the case of $p = \frac{1}{2}$.

In the next section we look at the rate of convergence for the simultaneous p-scaling algorithm. We conduct our analysis using a general p, but we remind the reader again to be aware that the overall algorithm convergence has only been proven for $p = \frac{1}{2}$.

3.4 Rate of Asymptotic Convergence

In this section we look at the theoretical rate of convergence for Algorithm 3.3.1 in exact arithmetic. We consider empirical examples of convergence in Chapter 6, as well as show examples of the effects of floating point arithmetic on the stability of the algorithm.

Theorem 3.4.1. *Let A be nonnegative, symmetric, and fully indecomposable, and let $f(x) = D(Ax)^{-p}D(x)^{1-p}e$, where $0 < p < 1$. If $x_{k+1} = f(x_k)$, with $x_0 = e$, and $x_\star$ is such that*

$P = D(x_\star)AD(x_\star)$ is a doubly stochastic matrix, and if $x_k = x_\star + d_k$, then

$$\lim_{k\to\infty} \frac{\|d_{k+1}\|}{\|d_k\|} \leq \|J_f(x_\star)\|$$

for any norm. Furthermore, for any $\epsilon > 0$ there exists a norm that depends on ϵ, $\| \star \|_{\rho,\epsilon}$ such that

$$\lim_{k\to\infty} \frac{\|d_{k+1}\|_{\rho,\epsilon}}{\|d_k\|_{\rho,\epsilon}} \leq \rho(J_f(x_\star)) + \epsilon.$$

Proof. We have that

$$x_{k+1} = f(x_k).$$

Substituting in $x_k = x_\star + d_k$ gives us

$$x_\star + d_{k+1} = f(x_\star + d_k).$$

We use the results of the linear map of f to obtain

$$x_\star + d_{k+1} = f(x_\star) + J_f(x_\star)d_k + \theta(d_k)$$

We know that $x_\star$ is a fixed point of the function f, and so we can simplify to

$$d_{k+1} = J_f(x_\star)d_k + \theta(d_k).$$

By taking the norm of both sides, using the triangle inequality, and using the compatibility of matrix and vector norms, we have as a result

$$\frac{\|d_{k+1}\|}{\|d_k\|} \leq \|J_f(x_\star)\| + \frac{\|\theta(d_k)\|}{\|d_k\|}.$$

If we take the limit of this result, then the $\theta(d_k)$ term will disappear, leaving

$$\lim_{k\to\infty} \frac{\|d_{k+1}\|}{\|d_k\|} \leq \|J_f(x_\star)\|.$$

The first result is proven. The second result follows from (Horn and Johnson [36] pg. 297). □

Now we want to compare the rates of convergence between the Sinkhorn-Knopp algorithm and the simultaneous p-scaling algorithm. Let us recall that for the Sinkhorn-Knopp algorithm and for the norm described in Horn and Johnson we have the following result:

$$\lim_{k\to\infty} \frac{\|d_{k+2}\|}{\|d_k\|} \leq |\lambda_2(P)|^2 + \epsilon.$$

We consider $p = \frac{1}{2}$ for the simultaneous p-scaling algorithm, as we have shown that this is guaranteed to converge. Using our previous results on the value of the Jacobian given in Theorem 3.4.1, we have the following result:

$$\lim_{k\to\infty} \frac{\|d_{k+1}\|}{\|d_k\|} \leq \frac{|1-\lambda_n(P)|}{2} + \epsilon.$$

We have already stated that if the consensus similarity matrix is nearly uncoupled, then $\lambda_2(P)$ is close to 1. We next show that $\lambda_n(P) \geq 0$. In the worst case of $\lambda_n(P) = 0$, the error in the simultaneous p-scaling will be bounded by $\frac{1}{2}$, much smaller than $\lambda_2(P)$ for nearly uncoupled matrices.

Lemma 3.4.2. *If S is a similarity consensus matrix, which is a similarity matrix formed by the adjacency matrices of clustering results, then S is positive semi-definite.*

Proof. Recall that $S = \sum C(i)$ where

$$C(i)_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are clustered together} \\ 0 & \text{if } i \text{ and } j \text{ are not clustered together} \end{cases}$$

There are k positive eigenvalues of $C(i)$, each corresponding to a cluster, and there are $n-k$ eigenvalues equal to 0. Consider the following example:

$$C = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

C is made up of k rank 1 sub-matrices. This guarantees that there are $n-k$ eigenvalues equal to 0. Each sub-matrix is an $n_i \times n_i$ sub-matrix of ones, which guarantees that C has an eigenvalue of n_i corresponding to each sub-matrix. Thus $C(i)$ is positive semi-definite, and S is positive semi-definite. □

When considering other similarity matrices, it is important to note that the cosine and Gaussian similarity matrices are also always positive semi-definite. The k-shared nearest neighbor matrix, however, is not guaranteed to be positive semi-definite.

As a final comment, we wish to note that for non-symmetric matrices the Sinkhorn-Knopp algorithm will always have a better bound for the error than a generalization of the simultaneous p-scaling to non-symmetric matrices.

CHAPTER 4

Initializations and Visualizations

In this chapter we introduce a method for visualizing the stochastic consensus clustering process and results. We argue that through visualization, a user has the ability to validate the clustering, as well as potentially improve the quality of clustering results.

While there are many data visualization techniques that already exist, the technique we specifically propose helps the user validate the output of the stochastic consensus clustering. We do not argue that this visualization should supplant traditional visualization approaches, but rather supplement them. The visualization method does not require extra calculations, and as such, it is achieved at little additional cost.

We also discuss the impact that initialization of the Markov chain can have on the stochastic clustering algorithm. We use the visualization method we propose in our discussions on initialization, and in the process show how the visualization can give insight to the user about initializations that perform poorly or well.

In order to illustrate examples we use the Ruspini data set [68], a set of points in $\mathbb{R}^2$. We also use the Ruspini data set for examples in Chapter 5 as well. The Ruspini data set can be seen in Figure 4.1.

We use the Ruspini data set purely for illustrative purposes, as we are able to see both the data as well as the stochastic consensus clustering output. In Chapter 6 we present more experimental results and show visualizations on a variety of different data sets that are considerably more complex.

We print here the stochastic clustering algorithm again in order to reduce page flipping for the reader. For the examples in this section, the consensus similarity matrix we use has been

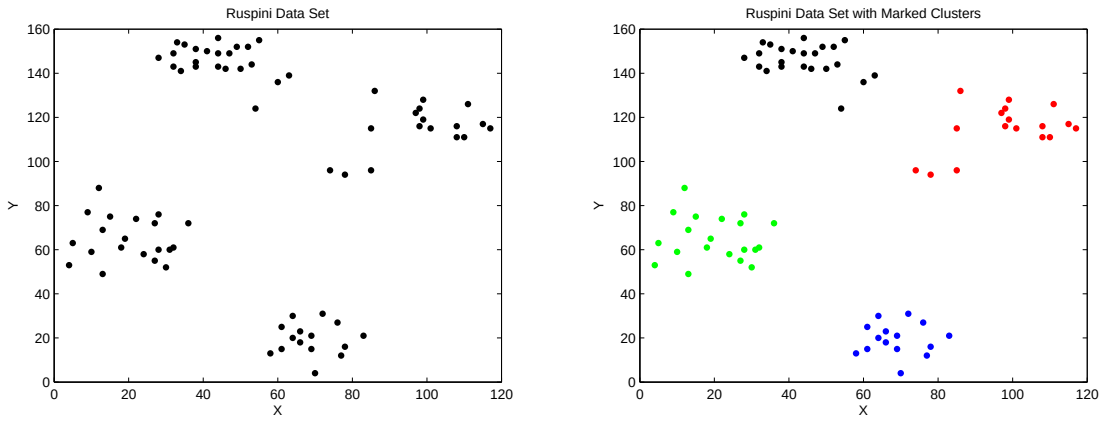

Figure 4.1: Ruspini data [left] and the Ruspini data separated into four clusters [right]

created using multiple runs of k-means. K-means is discussed in more detail in Chapter 5. We use the doubly stochastic matrix from the consensus matrix converted via the simultaneous p-scaling method discussed in Chapter 3.

Stochastic Consensus Clustering Algorithm [86]

Input: N observations $x_i \in \mathbb{R}^m$.

1. Create the consensus similarity matrix S using a clustering consensus of the user's choice.
2. Use matrix balancing to convert S into a doubly stochastic symmetric matrix P.
3. Compute eigenvalues of P. Sort the eigenvalues of P in descending order and find the largest difference between successive eigenvalues. Let k be the number of eigenvalues before the largest difference.
4. Create a random s_0^T.
5. Track the evolution $s_t^T = s_{t-1}^T P$. After each iteration, sort the elements of s_t^T and then separate the elements into k clusters by dividing the sorted list at the $k-1$ largest gaps. When this clustering remains the same for a user-defined number of iterations, the final clusters are determined.

Output: Clusters $C_1, \ldots, C_k$.

4.1 Visualization of the Evolving Markov Chain

Many visualization tools, like the ones mentioned in Chapter 1, are tools which attempt to provide the user with visualization either before or after a clustering algorithm has been run. Additionally some of these visualization methods require a large number of computations and scale poorly with the number of observations, such as the graph visualizations found in Gephi [8].

A very popular visualization method in use today is to project the data onto its first two or three principal components [52] [5] [31] [71]. Additional work has been done on using Kernel PCA [35] in the use of visualization [58] as well. However, PCA-based visualization methods are not a suitable visualization for validating the results of stochastic clustering because the stochastic clustering is relying on the interconnectedness of the data in the consensus matrix, as opposed to just relying on the locations of data points in the original space. We show examples of native PCA visualization using 2 principal components on data sets in Chapter 6.

Our proposed visualization with the stochastic clustering can work in two ways. The first way is that the user is returned a visualization upon completion of the algorithm. In this case the user has the ability to view the visualization in order to see that the expected results of the Simon-Ando theory are present. The second way is that the user is allowed to help determine the clustering in an interactive setting. In this way, the visualizations are presented before the final clustering has been determined. We have created a prototype user interface for Matlab that allows users to interact with the stochastic clustering algorithm in this way. We present a walk-through example in Appendix A.

We propose visualizing the steps of the Markov chain by plotting the indices of the probability vector, s, by the probability values: i.e. plotting i versus s_i. While plotting the probabilities of a Markov chain has always been possible, in a general Markov chain problem, watching these probabilities provides no extra information to the user. The reason for plotting the probabilities in stochastic clustering is due to the Simon-Ando theory underlying the algorithm. Recall that the short-run dynamics are exhibited by

$$\begin{aligned} \pi_t^T &\approx \left(\alpha_1 c_1^T \,\middle|\, \alpha_2 c_2^T \,\middle|\, \dots \,\middle|\, \alpha_k c_k^T \right) \\ &= \left(\tfrac{\alpha_1}{n_1}, \ \dots, \ \tfrac{\alpha_1}{n_1} \,\middle|\, \tfrac{\alpha_2}{n_2}, \ \dots, \ \tfrac{\alpha_2}{n_2} \,\middle|\, \dots \,\middle|\, \tfrac{\alpha_k}{n_k}, \ \dots, \ \tfrac{\alpha_k}{n_k} \right), \end{aligned}$$

and that the middle-run dynamics are exhibited by

$$\begin{aligned}\pi_t^T &\approx \left(\beta_1(t)c_1^T \;\middle|\; \beta_2(t)c_2^T \;\middle|\; \ldots \;\middle|\; \beta_k(t)c_k^T \right) \\ &= \left(\frac{\beta_1(t)}{n_1}, \;\ldots,\; \frac{\beta_1(t)}{n_1} \;\middle|\; \frac{\beta_2(t)}{n_2}, \;\ldots,\; \frac{\beta_2(t)}{n_2} \;\middle|\; \ldots \;\middle|\; \frac{\beta_k(t)}{n_k}, \;\ldots,\; \frac{\beta_k(t)}{n_k} \right).\end{aligned}$$

Throughout the short-run and middle-run dynamics, observations within a cluster should have approximately equal probability. We form clusters by grouping observations that have approximately the same probability together. When we plot the probability vectors, this yields a plot in which there are bands of probability at various values, corresponding to the different clusters.

Plotting the probabilities is a simple command in Matlab: *plot(1:n,s,'.')*. For further detail a user can also color the data points by the clustering determined from Step 5 of the stochastic clustering algorithm. An example of these probability plots can be found in Figure 4.2. Notice the bands of probability values around 4 distinct values. This is because there are four clusters in the Ruspini data set.

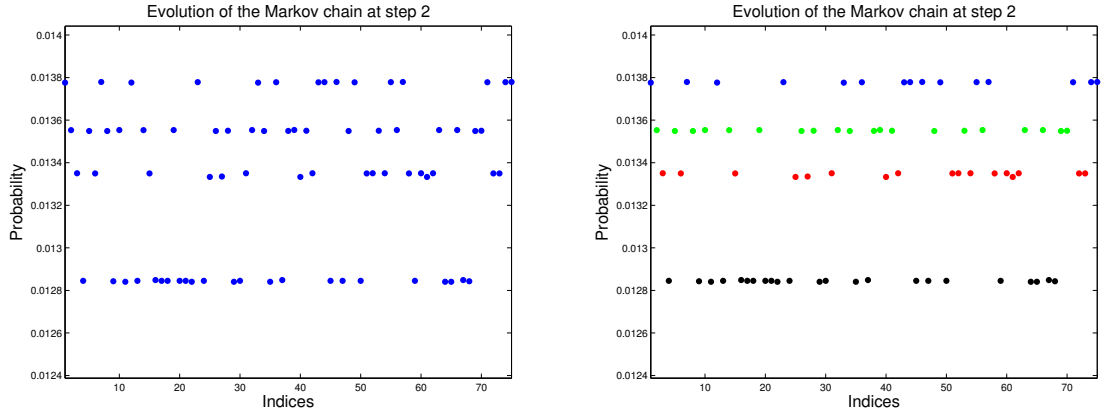

Figure 4.2: Plots of s_4, the 4th evolution of the Markov chain, on the Ruspini data. [Left] is a plot without coloring according to separating at the $k-1$ largest gaps, while [Right] shows coloring by determined cluster.

As the Markov chain evolves through time, we can track this evolution with a series of plots. From the theory of the stochastic clustering algorithm we know that the Markov chain should go through several distinct phases. We can see these phases in action and can see the probabilities for each state separate into bands according to which cluster the state belongs (Figures 4.3,

4.4, 4.5). If the Markov chain increments through time, these bands of probabilities then tend towards the uniform probability of $\frac{1}{n}$, or in the case of the Ruspini data: 0.01333.

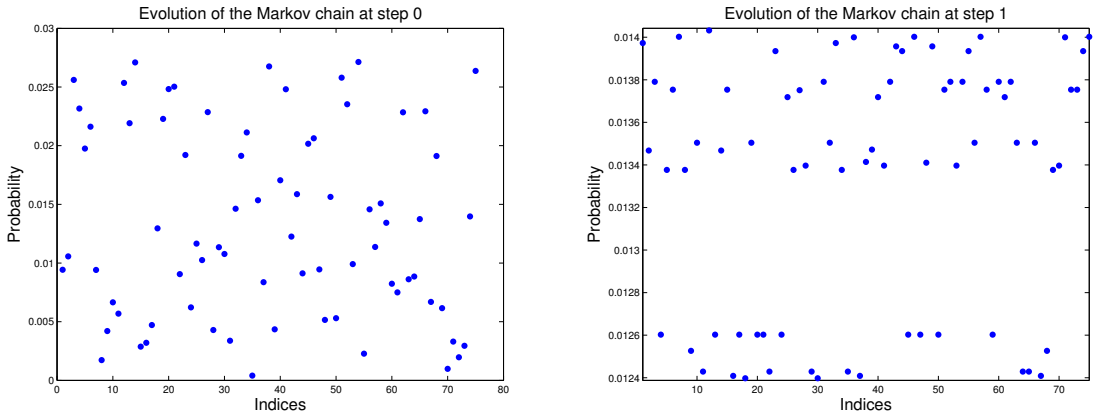

Figure 4.3: Plots of s_0 [Left] and s_1 [Right] on the Ruspini data.

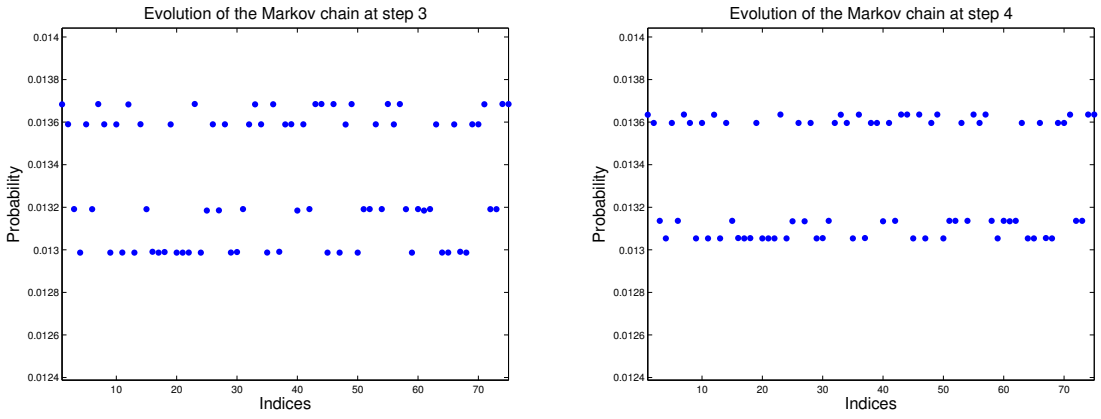

Figure 4.4: Plots of s_2 [Left] and s_3 [Right] on the Ruspini data.

For the purpose of validation, the stochastic clustering algorithm returns the plot of probabilities for the step of the Markov chain at which the process stops. With each observation

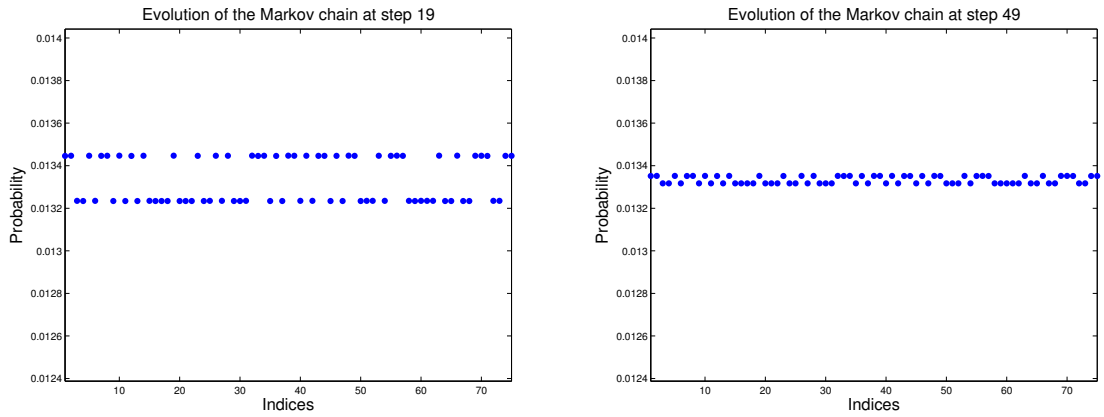

Figure 4.5: Plots of s_{19} [Left] and s_{49} [Right] on the Ruspini data. The probability vector is converging to the uniform probability vector.

assigned to a cluster it is not difficult to color the dots by cluster membership. This coloring allows us to easily verify if the probabilities form patterns as described by the Simon-Ando theory, and that the clustering captures these expected results. If we see a plot such as those in 4.3, then we know that we obtained a poor clustering.

The probability plots illustrate the idea of the Simon-Ando theory without a user needing to understand the underlying mathematics. The user only needs to know to look for bands of probability values, and gaps between. The simplicity in the visualization makes this concept a powerful tool even for novice data analysts.

Visualizations like this not only help us to better understand the resulting clustering, they can also help us in validating that the clustering algorithm returned relevant information. For example, if we see no bands of probabilities upon plotting, then we know that results of the stochastic consensus clustering are likely not useful.

For interactive use, we can plot each iteration of the Markov chain, step by step. Upon finding a plot of the probabilities that looks suitable, we can either specify probability values at which the clusters should be separated or separate the probability values as is normally done in stochastic clustering. While this is a less automated process, the results can be much better and are in line with the our motivating example of the data analyst from Chapter 1.

When visualizing the probability plots, it may seem natural to assume that values farther apart have less in common. This is a possibility, but it is can also be indicative of the initialization of the Markov chain. Each case may be different, and we warn about reaching conclusions on this aspect of the visualization.

4.2 Stochastic Clustering Initialization Concerns

We begin the discussion on initializations by revealing issues that can cause the stochastic clustering to either fail or perform in undesirable ways. The discussion of probability vectors leading to no solution paraphrases what has already been written in [86]. In the follow-up discussion on initial probability vectors leading to misleading solutions, we illustrate additional ways that poor initializations can cause problems in the stochastic clustering algorithm.

4.2.1 Initial Probability Vectors Leading to No Solution

We know from our discussion in Chapter 3 that the $n \times n$ doubly stochastic matrix P obtained is irreducible and primitive. Because of this we are guaranteed that the stationary distribution vector for P is

$$s^T = \begin{pmatrix} \frac{1}{n}, & \frac{1}{n}, & \ldots, & \frac{1}{n} \end{pmatrix}.$$

If we have chosen our initial probability vector to be the stationary vector, then it is obvious that no clustering results can be found, as the vector s will never change when multiplied into P. Additionally it can be shown that if we make a small perturbation such that

$$s_0^T = \begin{pmatrix} \frac{1}{n} + \epsilon_1, & \frac{1}{n} + \epsilon_2, & \ldots, & \frac{1}{n} + \epsilon_n \end{pmatrix}$$

where $\sum_{i=1}^{n} \epsilon_i = 0$ and $0 \leq \frac{1}{n} + \epsilon_i \leq 1$, then if the vector

$$\epsilon^T = \begin{pmatrix} \epsilon_1, & \epsilon_2, & \ldots, & \epsilon_n \end{pmatrix}$$

is in the null space of P, $s_0^T P = s$. That is to say, if the error vector is in the null space of P, s_t will converge to the stationary distribution after only one step of the Markov chain. Thus, if P is singular, then the Markov chain still has the potential to reach the uniform distribution before short-run or middle-run dynamics are witnessed. If the short-run or middle-run dynamics are skipped, this will prevent any clustering from being determined.

Depending on how the consensus matrix is constructed, it is not uncommon for it to be singular (see Lemma 4.2.1), while the cosine and Gaussian consensus matrices are very often non-singular.

Lemma 4.2.1. *If S is a consensus similarity matrix and $S_{ij} = S_{ii}$ for some $i \neq j$, then S is singular.*

Proof. Recall that S_{ij} is the number of times that objects i and j clustered together. If $S_{ij} = S_{ii}$, then objects i and j clustered together every time a clustering algorithm was run. This means

that $S_{ik} = S_{jk}$ for all $1 \leq k \leq n$. Since two rows of the matrix are exactly the same, then S is non full rank and is singular. $\square$

4.2.2 Initial Probability Vectors Leading to Misleading Solutions

A more dangerous situation can arise regardless of whether P is singular or non-singular. For this discussion let us assume that the states of P are ordered so that the first n_1 states correspond to cluster 1, the next n_2 states correspond to cluster 2, and so on. From our discussions in Chapter 2 we know that in the short-run dynamics

$$s_t^T \approx \left(\begin{array}{c|c|c|c} \alpha_1 c_1 & \alpha_2 c_2 & \dots & \alpha_k c_k \end{array} \right),$$

where each α_i is a constant that depends on the initial probability vector s_0^T. Because the short-run dynamics are dominated by the connections within the clusters, the we know that for any t during the short-run, the summation of probability within the clusters is approximately constant during the short-run.

$$\begin{aligned} \sum_{i=1}^{n_1} s_0(i) &\approx \sum_{i=1}^{n_1} s_t(i) \approx \sum_{i=1}^{n_1} \alpha_1 c_1 \\ \sum_{i=n_1+1}^{n_1+n_2} s_0(i) &\approx \sum_{i=n_1+1}^{n_1+n_2} s_t(i) \approx \sum_{i=1}^{n_2} \alpha_2 c_2 \\ &\vdots \\ \sum_{i=n-n_k+1}^{n} s_0(i) &\approx \sum_{i=n-n_k+1}^{n} s_t(i) \approx \sum_{i=1}^{n_k} \alpha_k c_k \end{aligned}$$

The summation of the elements of s_0 within a cluster can be thought of as the initial weight for each cluster. We know that each c_i is a uniform vector of length n_i, that is $c_i = (\frac{1}{n_i}, \frac{1}{n_i}, \dots, \frac{1}{n_i})$. It is obvious then that there may be a confounding issue between clusters i and j if $\alpha_i c_i = \alpha_j c_j$. This will occur if clusters i and j have a proportional weight based off of the number of states in each cluster. We can determine how the α_i and α_j are related based off of the size of each cluster.

$$
\begin{array}{ccc}
\alpha_i c_i & \approx & \alpha_j c_j \\
\left(\frac{\alpha_i}{n_i}, \quad \frac{\alpha_i}{n_i}, \quad \ldots, \quad \frac{\alpha_i}{n_i} \right) & \approx & \left(\frac{\alpha_j}{n_j}, \quad \frac{\alpha_j}{n_j}, \quad \ldots, \quad \frac{\alpha_j}{n_j} \right) \\
\frac{\alpha_i}{n_i} & \approx & \frac{\alpha_j}{n_j}
\end{array}
$$

Thus if the weight of cluster i is $\frac{n_i}{n_j}$ times the weight of cluster j, the states in clusters i and j will reach approximately the same values in the short-run stabilization. Even if only two clusters are close to the correct proportional weights, this can throw off the stochastic clustering process. Two separate clusters may accidentally be merged into one, while one cluster may be separated into two.

A further issue that may affect the results of the stochastic clustering algorithm is the rate at which each cluster in the doubly stochastic matrix demonstrates the short-run dynamics. If different clusters reach the short-run at different times, this may add additional confounding factors, even when the initial weights do not fall in the areas described above.

Figures 4.6 and 4.7 show an example in which two clusters have reached the same probability value during the short-run dynamics, due to appropriate initial weights. Because the largest gaps no longer occur between clusters of probabilities, a solution here would be misleading. Color for the clusters has been added to highlight this point.

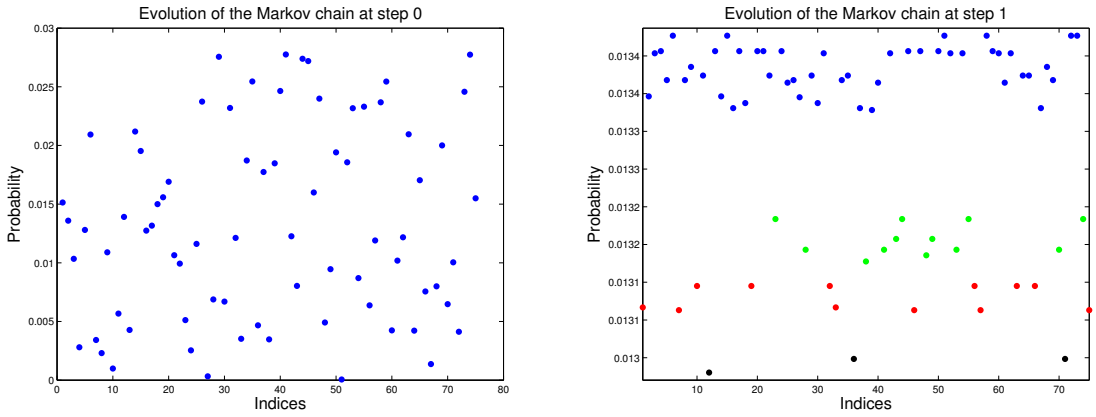

Figure 4.6: Plots of s_0 [Left] and s_1 [Right] on the Ruspini data.

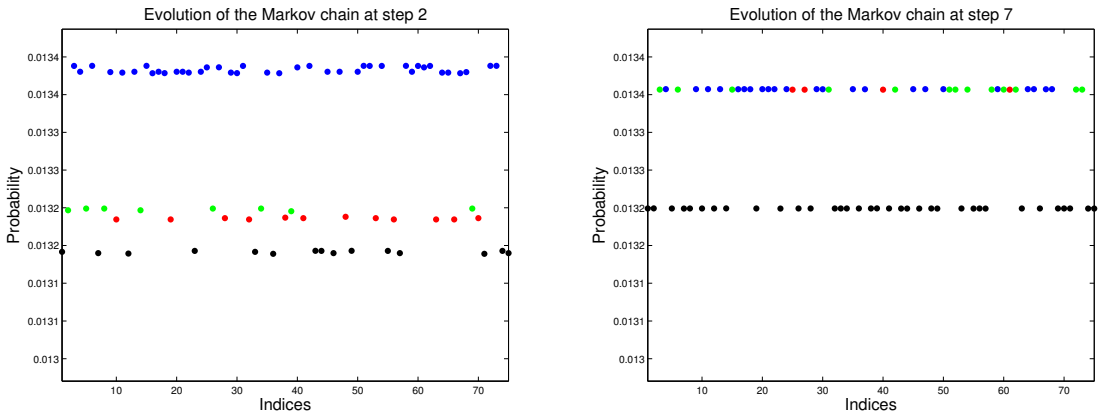

Figure 4.7: Plots of s_2 [Left] and s_7 [Right] on the Ruspini data. In s_2 we see that two of the clusters have reached approximately the same value before the short-run dynamics are witnessed. In s_7 we see another problem that if too many steps are taken some of the differences can become meaningless.

While we had to specify non-random initializations for the Ruspini data set, data sets with more observations, higher dimensionality, and less well-defined clusters are at risk of being subject to misleading clustering solutions. This leaves us in a particularly unfortunate place as we have no metrics to determine what a good initial probability vector is.

4.3 Using Multiple Initial Probability Vectors, and Extending Visualizations

An easy solution to a poor random initial probability vector is to just pick a different random initial probability vector. After we run the clustering, we can look at the visualization, and then restart if the solution does not look acceptable. As only a small number of matrix vector multiplications are required for a restart of the Markov chain, this is not a very costly method. In most cases of poor initialization that we encountered throughout this research for the stochastic clustering algorithm, we merely restarted the Markov chain.

An alternative is to use two initial probability vectors from the start. Thus instead of watching $s_t^T = s_{t-1}^T P$ only, we instead watch both the evolution of s and the evolution of a second probability vector $r_t^T = r_{t-1}^T P$. Instead of representing each observation as its single probability $s_t(i)$ at each step of the Markov chain, we instead have a pair of probabilities for each observation: $(s_t(i), r_t(i))$. Thus each observation is an ordered pair of probabilities. We

do not lose the ability to visualize our data with a second initial probability vector, as our visualization method is now to plot one probability vector versus the other.

When considering the Markov chain with only one probability vector, we know that observations in the same cluster will all have approximately equal probability after the Markov chain has reached the short-run equilibrium. In the case of considering two initializations we know that each probability vector as it steps through the Markov chain will exhibit short-run dynamics. However, a given cluster may reach the short-run at a different probability value between the two probability vectors, due to different initializations. In this case when we plot two probability vectors we expect to see related observations clump together at distinct points in the plane, and then eventually converge to $(\frac{1}{n}, \frac{1}{n})$.

In Figure 4.8 and Figure 4.9 s_0 is the same initial probability vector used in section 4.2.2, while r_0 is a second initial probability vector. We can see that the separation of clusters is clear even though s_0 was previously considered a poor choice. In the later steps, many of the observations in the same cluster appear collocated. This is analogous to the bands of probabilities that we saw in the case of using only one initial probability vector.

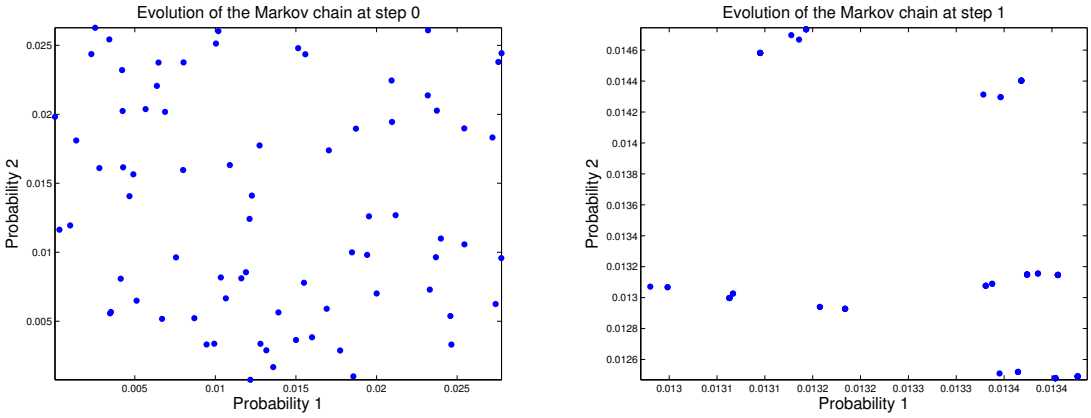

Figure 4.8: Plots of s_0 versus r_0 [Left] and s_1 versus r_1 [Right] on the Ruspini data.

We want to take a moment to discuss in further detail the graphs of Figure 4.9. Two of the clusters have nearly the same x-axis values. The x-axis corresponds to the vector s_2, and these values can be seen in Figure 4.7. However, because we added a second probability vector, we created space along the y-axis, effectively separating the two confounded clusters.

While there is no guarantee that using two probability vectors will prevent all issues of

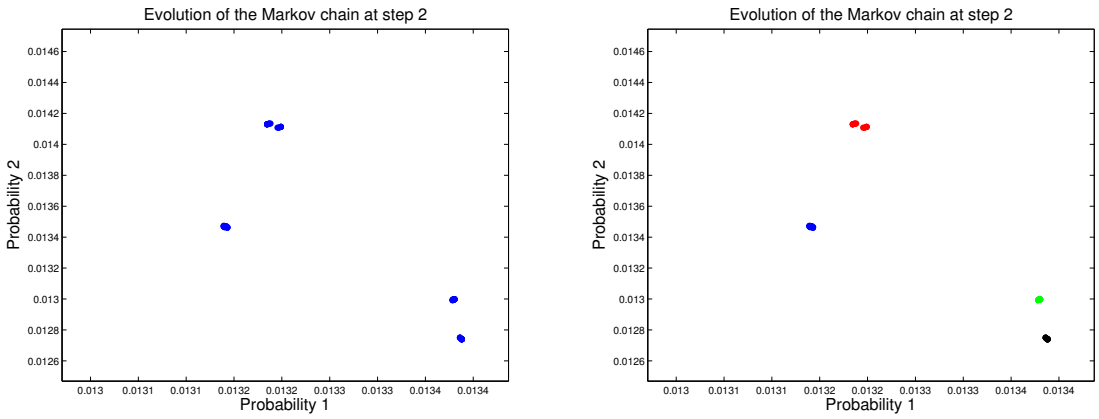

Figure 4.9: Plots of s_2 versus r_2 [Left] and s_2 versus r_2 colored after clustering [Right] on the Ruspini data.

misleading clusterings, it distinctly lowers the chances, as both initial probability vectors s and r would need to confound clusters in a similar way for them to remain confounded in two dimensions.

An important note is that due to using two probability vectors we can no longer cluster based off of the largest gap in probabilities. Instead we propose to use k-means clustering as the clusters are relatively compact, and spherical in nature. We do not require k-means, as any clustering algorithm that scales well with the number of observations should work well, but k-means is also relatively well-known and simple to implement.

We include this extra visualization method as an option in the user interface which we discuss in Appendix A. In addition, we supply an example in Appendix A to add clarity on how a user can interactively use the stochastic clustering algorithm.

CHAPTER 5

Experimental Setup

In this chapter we provide a background of the various algorithms that we use in the experiments section. We give a background of the k-means clustering method and the spectral clustering method. K-means is used both in the creation of consensus similarity matrices, as well as part of the spectral clustering algorithm. Spectral clustering is a very popular clustering method (over 1800 citations of [84] and 2000 citations of [72]) that is similar to the stochastic consensus clustering method. Like the stochastic consensus clustering, spectral clustering uses similarity matrices and can provide the user with k, the number of clusters in the data.

We do not go into any in-depth analysis of the theory behind k-means or spectral clustering. Instead we seek to provide the user with a general understanding, as well as references in case the reader wishes to follow up with more detail. A well-summarized background of the k-means algorithm and spectral clustering can also be found in [63] in pages 16-20 and pages 27-35.

5.1 K-means

K-means is a clustering algorithm that was designed to assign points $x_1, x_2, \ldots, x_n$ in $\mathbb{R}^m$ to clusters $C_1, C_2, \ldots, C_k$. The term k-means was first used in (MacQueen [48]), but other early work was done in (Hartigan [33], Lloyd [47], and Forgy [28]). The most widely recognized version of k-means seeks to minimize

$$\sum_{i}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|_2^2,$$

where μ_i is the mean of the points in C_i. Due to the fact that finding the optimal assignment is NP-hard (Mahajan et. al [49] and Aloise et. al. [1]), the k-means algorithm is an iterative process that converges to a local minimum of the objective function.

Algorithm 5.1.1. *Kmeans*

Input: *N observations $x_i \in \mathbb{R}^m$, and a number k of clusters to construct.*

- *Initialize with k clusters.*
- *Calculate the means: $\mu_i = \frac{1}{n_i} \sum_{x_j \in C_i} x_j$, where n_i is the number of points in C_i*
- *Update the clusters: For each x_j assign to C_i, where μ_i is the closest mean to x_j in Euclidean distance.*

Output: *Clusters $C_1, \ldots, C_k$.*

The initialization step has been left general as there are many different ways in which k-means can be initialized. A discussion on some initialization methods follows in section 5.1.1 and some k-means variants follows in section 5.1.2.

5.1.1 K-means Initializations

While there are many initializations for k-means we limit our discussion to three well-known initialization schemes to provide the reader with a general flavor. Examples of the initializations are also provided in figures 5.1 and 5.2.

- Random initialization - This initialization scheme randomly assigns each data point to one of the k clusters. The centroids formed by these initial clusters tend to be closely packed in the center of the data.
- Forgy initialization (Forgy [28]) - This initialization scheme picks k data points at random to be the k initial centroids. These centroids tend to be spread further apart than the centroids formed in the random initialization.
- K-means++ (Arthur and Vassilvitskii [4]) - This initialization method seeks to spread apart the initial centroids. For the specifics of step 3) we recommend the reader to the cited paper. The general steps are as follows:
    1. Pick a data point at random, this will be the first centroid.

2. Find the distance from all data points to all previously determined centroids.
3. Assign a probability to each data point based off of the distances, and pick a data point from the distribution - this is the next centroid.
4. Repeat 2) and 3) until k centroids have been chosen.

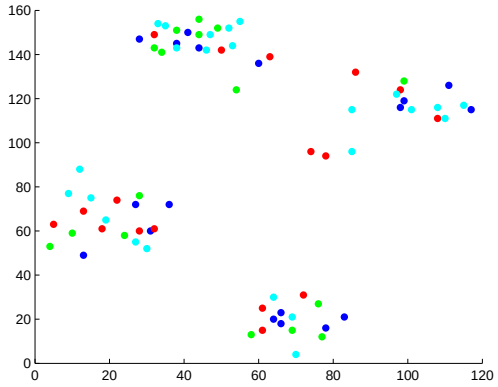

Figure 5.1: Example of the random initialization on the Ruspini data set.

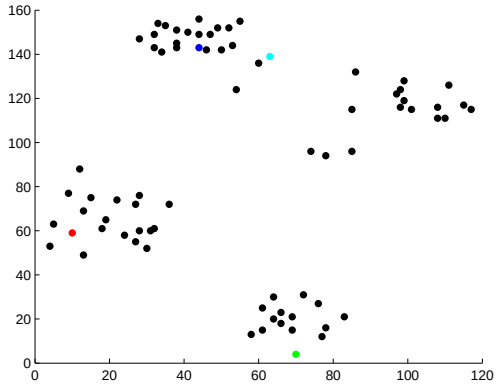

Figure 5.2: Example of the Forgy initialization on the Ruspini data set. Black data points do not yet belong to any cluster.

5.1.2 K-means Variants

In addition to the many initialization methods, k-means also has a plethora of variants. Again we seek only to provide the reader with a brief overview of some of the well known variants with references.

- Fuzzy k-means (Bezdek [11]) - A version of k-means in which each data point has a weight assigned to it for each cluster. The centroids are formed by the weighted averages of all the data points. The final result is a set of weights for each data point that indicates partial cluster assignment.
- Spherical k-means (Dhillon et. al. [23]) - In this version of k-means, instead of assigning x_i to cluster j based off of the minimum $\|x_i - \mu_j\|_2$, the assignment is done based off the maximum $\cos(\theta_{ij}) = \frac{x_i^T \mu_j}{\|x_i\|_2 \|\mu_j\|_2}$, the cosine similarity. The centroids are found by normalizing the average of the data points assigned to each cluster.

  Spherical k-means has been shown to be very good for directional data, especially textual data.
- K-means with other norms - The 2-norm $\| \star \|_2$ can be replaced with any other norm. During the cluster assignment minimize the desired norm, as opposed to the 2-norm.

5.2 Spectral Clustering

The name **spectral clustering** comes from being an algorithm which has its background in spectral theory - theory on eigenvalues and eigenvectors. Spectral clustering focuses on the eigenvalues and eigenvectors of a matrix referred to as the **graph Laplacian**. This is not a unique term, however, as there are many graph Laplacians. A comprehensive review of spectral clustering is given by von Luxburg [84].

Spectral clustering in its theoretical background is related to the *min-cut* and *n-cut* problems, though an understanding of these problems is not vital to understand the steps of the algorithm. We first give the definition of the graph Laplacian.

Definition 5.2.1. *The **unnormalized graph Laplacian** matrix is defined as*

$$L = D - S$$

where S is a similarity matrix, and D is a diagonal matrix such that $D_{ii} = \sum_j S_{ij}$.

For in depth theory on the Laplacian, including its relation to the Laplace-Beltrami operator we recommend the reader to [16]. We introduce two popular normalizations for the Laplacian below.

Definition 5.2.2. *The **normalized Laplacian** matrix is defined as*

$$\mathcal{L} = I - D^{-1/2} S D^{-1/2}$$

where S is a similarity matrix, and D is a diagonal matrix such that $D_{ii} = \sum_j S_{ij}$.

Definition 5.2.3. *The **normalized random walk Laplacian** matrix is defined as*

$$L_{rw} = I - D^{-1} S$$

where S is a similarity matrix, and D is a diagonal matrix such that $D_{ii} = \sum_j S_{ij}$.

The normalized Laplacian is used in a spectral clustering variant [59], as well as in Algorithm 5.2.4, while Shi and Malik [72] propose using the random walk Laplacian for use in Algorithm 5.2.4. It is not the goal of this thesis to investigate the merits of one Laplacian normalization over another, and as such we use the unnormalized Laplacian for experiments in Chapter 6.

It is know that L, L_{rw}, and $\mathcal{L}$ are positive semi-definite (Mohar [56] [57] and Chung [16]), and that $\lambda_n(L) = \lambda_n(L_{rw}) = \lambda_n(\mathcal{L}) = 0$, which corresponds to the eigenvector e.

Algorithm 5.2.4. *Spectral Clustering*

Input: *A similarity matrix S, and a number k of clusters to construct.*

1. *Create L, L_{rw}, or $\mathcal{L}$.*
2. *Compute the first k eigenvectors of the chosen Laplacian, $u_1, u_2, \ldots u_k$.*
3. *Construct $U \in \mathbb{R}^{n \times k}$ where the jth column of U is u_j.*
4. *For $i = 1, \ldots, n$ let $y_i \in \mathbb{R}^k$ be the ith row of U.*
5. *Cluster the points y_i into k clusters using k-means.*

Output: *Clusters $C_1, \ldots, C_k$.*

An alternative to having the user input a number k is to compute the eigenvalues of L (or L_{rw}, $\mathcal{L}$), and sort by value. The number of eigenvalues "near" 0, as determined by the largest gap, can be used as an automatically determined k.

5.2.1 Similarities Between Spectral Clustering and Stochastic Consensus Clustering

In Chapter 6 we make experimental comparisons between spectral clustering and stochastic consensus clustering. Before we do, there are several areas of particular interest in which these two algorithms have in common. While we do not know what these commonalities imply, we feel it is important to point them out before comparing them experimentally.

- The apparent similarities are that both spectral clustering and stochastic consensus clustering are graph-based in nature, that both rely on the use of a similarity matrix.
- Both the consensus similarity matrix and the graph Laplacian are positive semi-definite.
- For finding the number of clusters in stochastic consensus clustering, we look to the number of eigenvalues "near" 1, as determined by the largest gap. For finding the number of clusters in spectral clustering we look to the number of eigenvalues "near" 0, as determined by the largest gap.

  This is particularly interesting because the arguments behind each hinge on similar concepts. Additionally, the eigenvector corresponding to $\lambda_1(P)$ and to $\lambda_n(L)$ is e, the vector of all ones.

5.3 Experimental Setup

For all experiments the consensus similarity matrix is created by using multiple runs of the k-means algorithm using the random initialization. We only let the k-means algorithm run 10 iterations instead of allowing it to converge, as we want both increased variation of clustering results to use in consensus, and we want to bound the number of computations required.

We separate our experiments into two categories: experiments on data sets with known clusters, and experiments on data sets for exploration. Experiments on the former group provide results corresponding to the efficiency of the simultaneous scaling algorithm as compared to the Sinkhorn-Knopp algorithm; the validity of using other similarity matrices for stochastic clustering; and the comparison of stochastic clustering to spectral clustering. Experiments on data sets for exploration show how stochastic consensus clustering can make meaningful clusters, even when we have no prior knowledge of what the "true" clusters are.

In both cases we show examples of the visualization of the stochastic consensus clustering results. We see that the plots generated by the stochastic consensus clustering clearly show where clusters are, and whether the clustering results are poor. In the experiments in which we know the true clusterings, we also show the data projected onto the principal components as documented in [52] and [31].

In Chapter 3 we introduced the simultaneous scaling algorithm and showed that in the limit, the error is smaller than the Sinkhorn-Knopp algorithm when applied to nearly uncoupled similarity matrices. In Chapter 6 we compute the error per iteration of both the simultaneous scaling and the Sinkhorn-Knopp algorithms, using the infinity norm. The error is computed by $\|x_k - x_\star\|_2$. The number of operations for the simultaneous scaling is n^2+2n, while the Sinkhorn-Knopp requires $n^2 + n$ operations. We feel it is acceptable to compare the per iteration error, as each method is $O(n^2)$ per iteration.

The theorems and results in Chapter 3 were also discussed in exact arithmetic. We show experiments showing the difference per iteration between double precision and quadruple precision calculations of the simultaneous scaling algorithm, in order to experimentally validate the algorithm in finite arithmetic. We compute the absolute error, $\|x_k^d - x_k^q\|_2$, where x_k^d is the kth iteration of the absolute scaling in double precision and x_k^q is the kth iteration of the absolute scaling in quadruple precision. We compute the relative error as $\frac{|x_k^d - x_k^q\|_2}{\|x_k^q\|_2}$. These calculations are computed in quadruple precision.

In order to determine the validity of other similarity matrices, we plot the largest magnitude eigenvalues of the associated doubly stochastic matrix. We know that in order for probability values to appear in bands, the Simon-Ando theory requires a nearly uncoupled matrix. If the second largest eigenvalue is not near 1, then the similarity matrix is not nearly uncoupled (by Theorems 3.1.8 and 3.1.9), and is unsuitable for stochastic clustering, as the Simon-Ando theory will not apply.

In order to numerically compare results between spectral clustering and consensus clustering, we need a measure of wellness for clustering. We use the accuracy metric, sometimes called the purity metric ([51] page 329), which is equal to the ratio of the number of correctly clustered observations to the total number of observations. We use maximal matching to assign each created cluster to a target cluster.

CHAPTER 6

Experimental Results and Discussion

In this chapter we present the results of the experiments discussed in Chapter 5. We first show results on data sets with known cluster structure. In these examples we show that the simultaneous scaling algorithm is computationally more efficient than the Sinkhorn-Knopp algorithm when balancing consensus similarity matrices. In addition, we show the relative and absolute difference between double precision and quadruple precision calculation of the simultaneous scaling algorithm.

We examine the stochastic consensus clustering algorithm on several data sets with high accuracy. These data sets were chosen to come from different application areas to show the stochastic consensus clustering on a breadth of data types. We include the visualization plots and report the clustering accuracy, which enables us to see the separation in the probability values, as opposed to blindly trusting in the cluster output.

We then examine data sets for which we have no clustering solution using the stochastic consensus clustering algorithm as an exploratory tool. These examples are meant to demonstrate how even in situations in which the clusters are unknown, the visualization can still impart useful information to the user. In these cases we cluster the data, though there is no correct answer with which we can verify our results. Instead we must use a subjective validation based on cluster output and visualization. This scenario for clustering occurs widely in industry, and is noted in our motivating example in Chapter 1. Often times clusters are created this way as a preliminary step for creating classification models on each cluster.

6.1 Clustering Data Sets with Known Cluster Structure

In this section we look at three examples of data sets with known number of clusters and known cluster membership. We use data in which the observation counts vary from hundreds, to thousands, to ten thousand. In addition the domain of each data set is distinctly different, as well as the number of attributes for each data set.

6.1.1 Breast Cancer Data Set [50]

This data set from the UCI data repository [7] contains 699 observations and 9 dimensions. Each observation represents a tissue sample, and the attributes are measurements taken of the tissue sample from lab analysis. Each tissue sample is categorized as either benign or malignant. There are two clusters in the data.

Figure 6.1 shows the error per iteration of the Sinkhorn-Knopp matrix balancing as compared to the simultaneous scaling method with $p = 1/2$ and $p = 2/3$. We attribute the odd behavior of the error in the range below 10^{-15} to calculations involving values around the size of machine epsilon. We note the slow convergence of the Sinkhorn-Knopp algorithm, and estimate that it would take around 700 iterations before an error of 10^{-10} is reached using the Sinkhorn-Knopp algorithm.

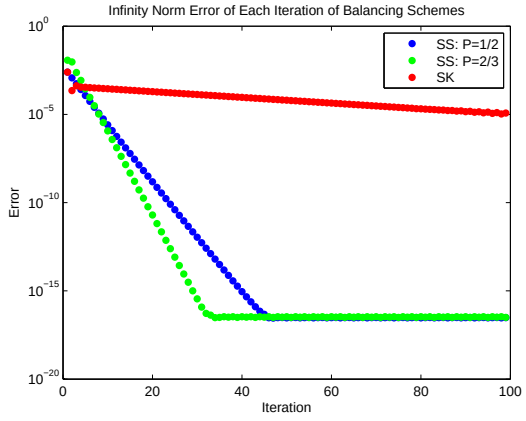

Figure 6.1: Plot of the error per iteration of various balancing schemes on the breast cancer data set.

Figure 6.2 shows the error at each iteration of the simultaneous scaling algorithm between double precision and quadruple precision. The error calculations were computed in quadruple

precision, and we can see that the absolute and relative errors converge at less than 10^{-14}.

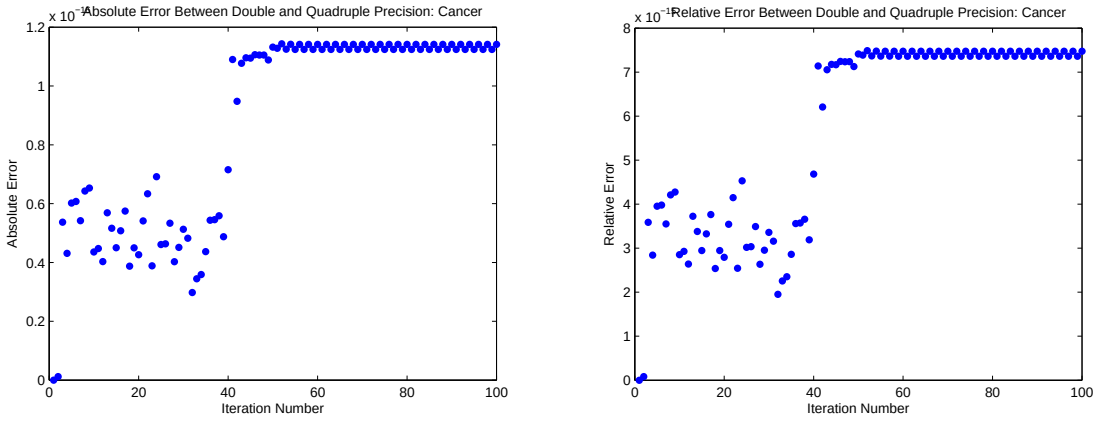

Figure 6.2: Absolute (left) and relative (right) error between double and quadruple precision of the simultaneous scaling on the breast cancer data set, computed in quadruple precision.

Figure 6.3 shows the eigenvalues of the doubly stochastic matrix created for the four different similarity matrices. We see that only the consensus similarity and the Gaussian similarity matrices have the desired eigenvalue spread. Both have the second largest eigenvalue close to 1. While the k-nearest neighbor similarity matrix also has the second largest eigenvalue near 1, it has no break in eigenvalues, indicating that a lack of cluster structure.

As only the Gaussian and consensus similarity matrices have the necessary structure for a successful outcome in using stochastic clustering, we examine accuracy and timing results for just these matrices. We use the stochastic clustering method and spectral clustering on both the consensus and Gaussian matrices. The results are included in Table 6.1. In this table we see that both spectral clustering and stochastic clustering perform well using either similarity matrix. As the number of observations is small, the timing differences between construction of the similarity matrices is very small.

Figure 6.4 shows the visualization of the clustering as determined by the stochastic consensus clustering algorithm. We see that there is a visible separation in probability values, and thus feel confident that two distinct clusters are represented. We also show the same plot ordered by cluster membership. While unknown data will never be ordered this way, we present the picture to confirm that each band of probabilities does indeed correspond to a cluster.

We can compare the visualization of the clustering in Figure 6.4 to the visualization of the data set using Principal Components Analysis (PCA) in Figure 6.5. In this case, PCA yields a

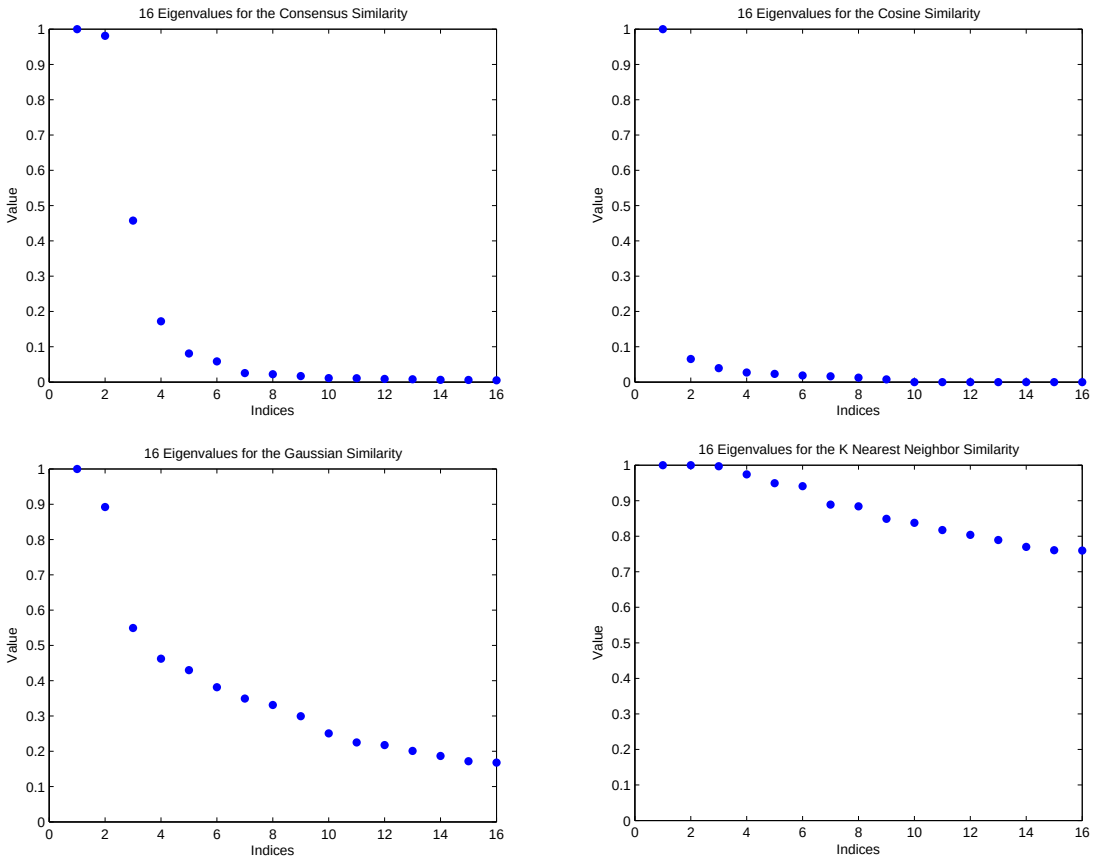

Figure 6.3: Eigenvalue plots for similarity matrices on the breast cancer data set.

Table 6.1: Accuracy and timing results for the breast cancer data set.

	Gaussian Similarity Matrix	Consensus Similarity Matrix
Construction Time	0.8 sec	2.1 sec
Spectral Clustering Accuracy	0.927	0.9585
Stochastic Clustering Accuracy	0.9642	0.9585

visualization in which we can see the two clusters.

6.1.2 Medlars, Cranfield, Cisi (MCC) Data Set ([22] pg. 74)

This data set is a conglomeration of three smaller data sets: Medlars, Cranfield, and Cisi. Each of these data sets are article abstracts from a given field.

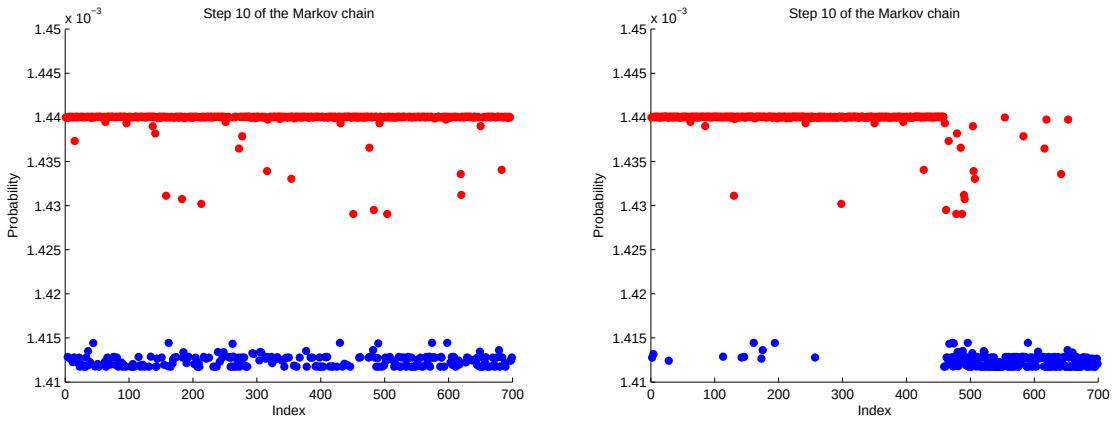

Figure 6.4: Visualization of the clustered data using the consensus similarity matrix on the breast cancer data set. Indices are unordered (left) and ordered by true cluster (right).

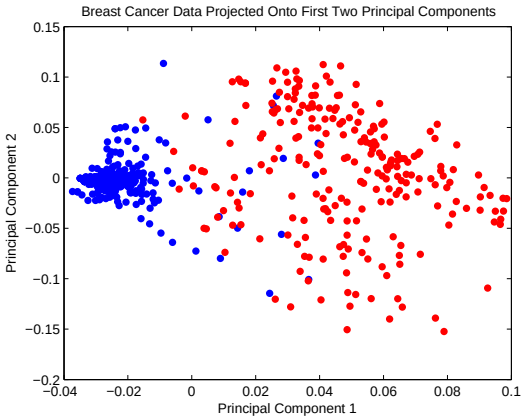

Figure 6.5: Visualization of the raw breast cancer data using PCA. The observations have been colored by cluster membership.

- Medlars: 1033 medical abstracts
- Cranfield: 1398 aerodynamics abstracts
- Cisi: 1460 information science abstracts

In total there are three main clusters and a total of 3891 documents. The set contained 15864 unique terms. However, after removing terms that do not appear in multiple documents (as these can provide no clustering information) the number of terms is slightly lower.

Figure 6.6 shows the error per iteration of the Sinkhorn-Knopp matrix balancing as compared to the simultaneous scaling method with $p = 1/2$ and $p = 2/3$. We again note the odd behavior when the error has dropped below 10^{-15}, and attribute this to the fact that machine epsilon for double precision numbers is around 10^{-16}.

We note again that the Sinkhorn-Knopp algorithm exhibits slow convergence, though not nearly as slow as in Figure 6.1. This corresponds with the fact that the second largest eigenvalue of the balanced consensus matrix is lower for the MCC data than the breast cancer data. We see in this case that it takes fewer than 100 iterations before an error of 10^{-10} is reached using the Sinkhorn-Knopp algorithm.

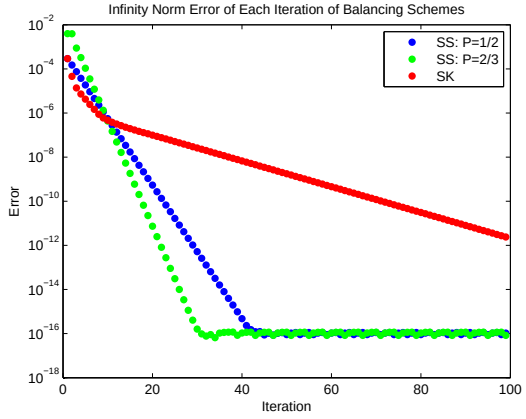

Figure 6.6: Plot of the error per iteration of various balancing schemes on the MCC data set.

Figure 6.7 shows the error at each iteration of the simultaneous scaling algorithm between double precision and quadruple precision. The error calculations were computed in quadruple precision, and we can see that the absolute and relative errors converge at less than 10^{-13}.

Figure 6.8 shows the eigenvalues of the doubly stochastic matrix created for the four different similarity matrices. We see that only the consensus similarity and the Gaussian similarity matrices have the desired eigenvalue spread, though the Gaussian similarity matrix is more suspect due to the fact that the second largest eigenvalue is farther away from 1.

We use the stochastic clustering and spectral clustering on both the consensus and Gaussian matrices. The results are included in Table 6.2. In this table we see that using the Gaussian similarity resulted in poor clusterings for both spectral clustering and stochastic clustering. On the other hand, stochastic clustering performed very well in comparison to spectral clustering

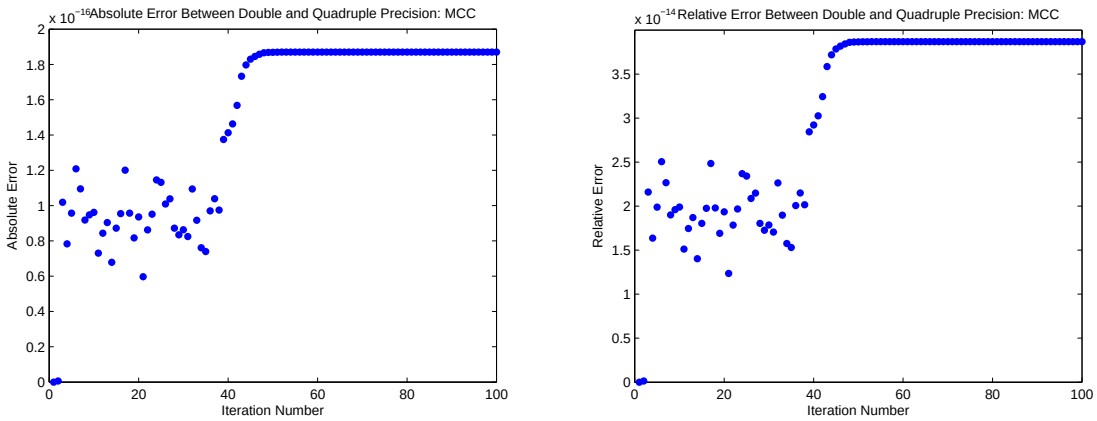

Figure 6.7: Absolute (left) and relative (right) error between double and quadruple precision of the simultaneous scaling on the MCC data set, computed in quadruple precision.

when using the consensus similarity matrix.

The timing results also show that the consensus similarity matrix was faster to compute than the Gaussian similarity matrix. This is due to the fact that the Gaussian similarity requires pairwise comparisons of observations.

Table 6.2: Accuracy and timing results for the MCC data set.

	Gaussian Similarity Matrix	Consensus Similarity Matrix
Construction Time	383 sec	257 sec
Spectral Clustering Accuracy	0.5623	0.6600
Stochastic Clustering Accuracy	0.3755	0.9797

Figure 6.4 shows the visualization of the clustering as determined by the stochastic consensus clustering algorithm. We see a visible banded structure in the probability values, indicating three main clusters. As an interesting side note, we see that the green dots are separated into two bands, which could indicate a possible sub-cluster in one of the document sets.

We can compare the visualization of the clustering in Figure 6.9 to the visualization of the data set using (PCA) in Figure 6.10. In the visualization using PCA it is not easy to see separation between the three clusters.

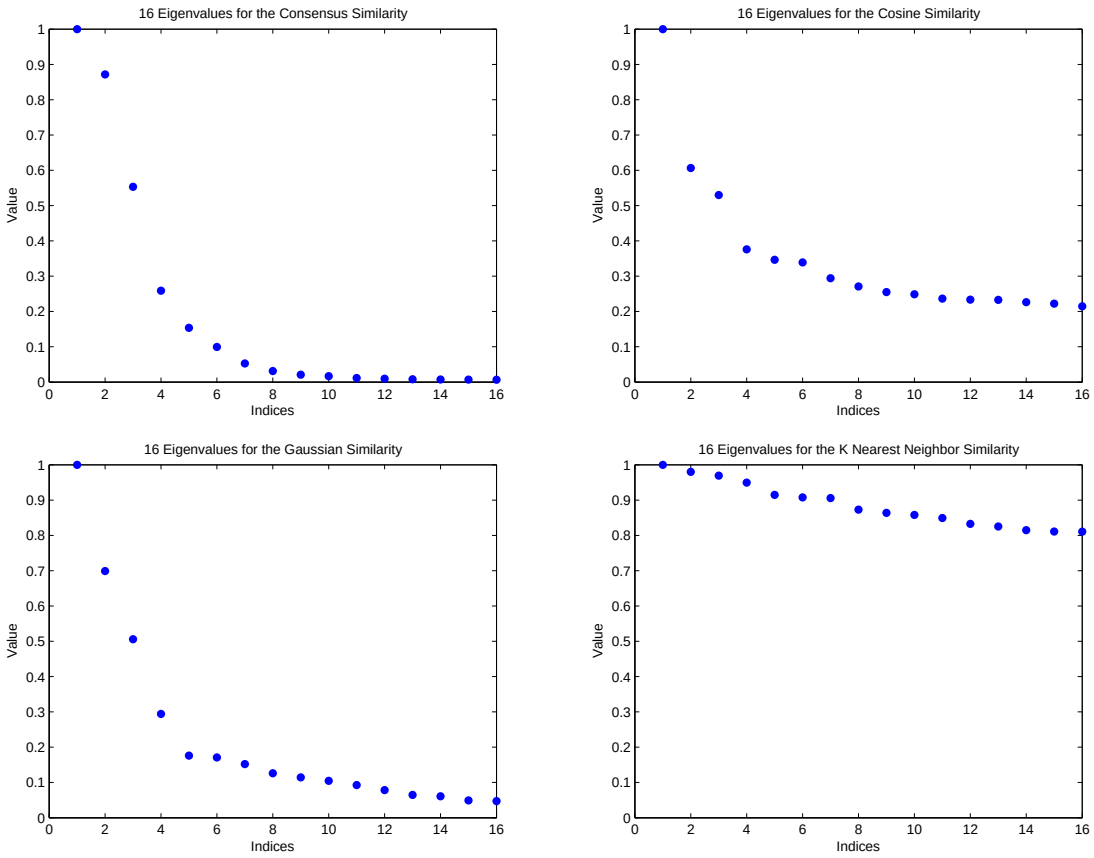

Figure 6.8: Eigenvalue plots for similarity matrices on the MCC data set.

6.1.3 Digits Data Set from the MNIST Database [44]

This is a data set from the National Institute of Standards and Technology in which each observation is a handwritten number. The images have been converted to pixels, and then size-normalized and centered. We use a subset of the data set which contains only images of the numbers 1, 2, and 7. This subset contains 13262 observations with three main clusters corresponding to the numbers 1, 2 and 7. There are 784 dimensions which correspond to the pixels in the 28-by-28 pixel grid.

Figure 6.11 shows the error per iteration of the Sinkhorn-Knopp matrix balancing as compared to the simultaneous scaling method with $p = 1/2$ and $p = 2/3$. We continue to ignore the results of error in the range of machine epsilon, as the maximum precision has already been reached.

As in previous examples, the Sinkhorn-Knopp algorithm exhibits slower convergence than

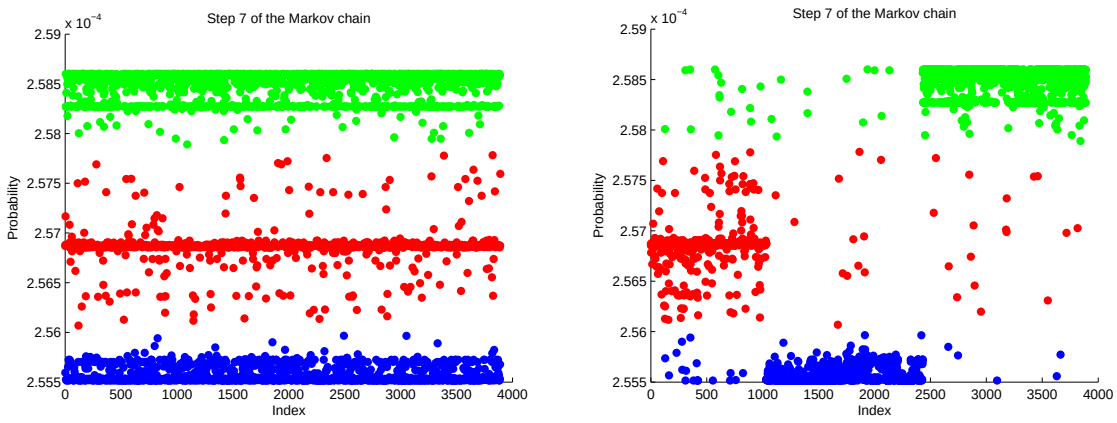

Figure 6.9: Visualization of the clustered data using the consensus similarity matrix on the MCC data set. Indices are unordered (left) and ordered by true cluster (right).

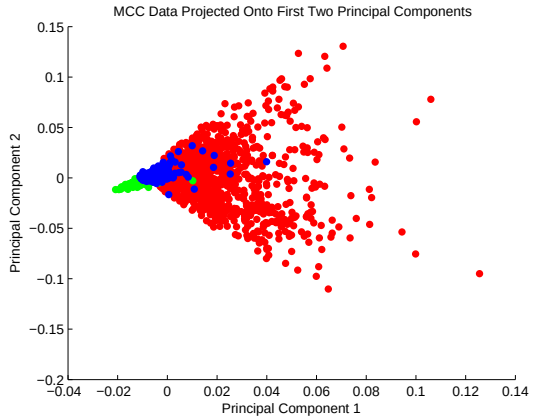

Figure 6.10: Visualization of the raw MCC data using PCA. The observations have been colored by cluster membership.

the simultaneous scaling for creating the doubly stochastic matrix, P. We again see the correlation between the largest eigenvalue and the slow convergence of the Sinkhorn-Knopp algorithm.

Figure 6.12 shows the error at each iteration of the simultaneous scaling algorithm between double precision and quadruple precision. The error calculations were computed in quadruple precision, and we can see that the absolute and relative errors converge at less than 10^{-12}.

Figure 6.13 shows the eigenvalues of the doubly stochastic matrix created for the four different similarity matrices. The consensus similarity matrix again yields eigenvalues that are

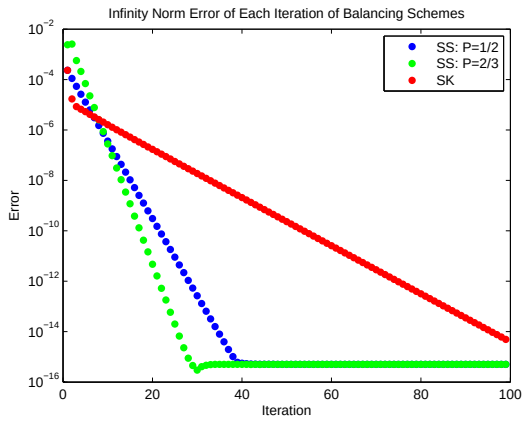

Figure 6.11: Plot of the error per iteration of various balancing schemes on the digits data set.

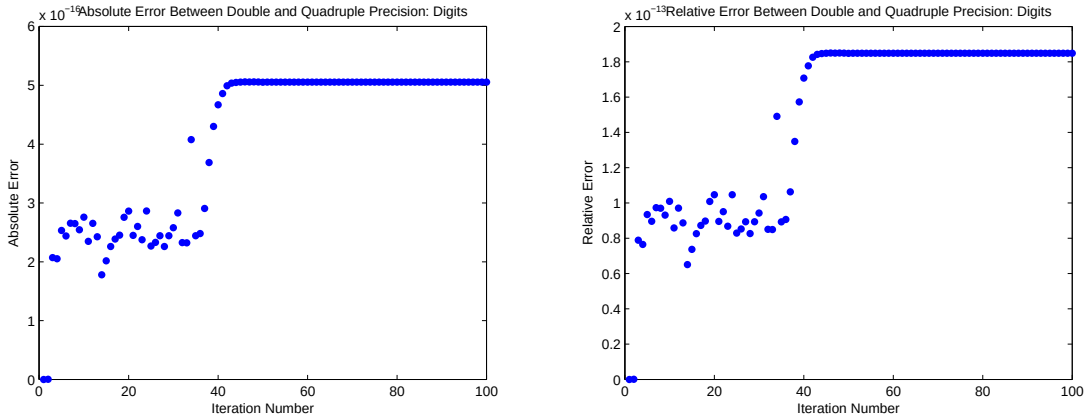

Figure 6.12: Absolute (left) and relative (right) error between double and quadruple precision of the simultaneous scaling on the Digits data set, computed in quadruple precision.

most in line with what is necessary for stochastic clustering to ensure a nearly uncoupled matrix.

We use the stochastic clustering method and spectral clustering on both the consensus and Gaussian matrices. The results are included in Table 6.3. As was the case with the MCC data set, we see that using the Gaussian similarity matrix resulted in poor clusterings for both spectral clustering and stochastic clustering. The consensus matrix clustered well in both cases of spectral and stochastic clustering.

The timing results also show that the consensus similarity matrix was faster to compute than the Gaussian similarity matrix. We see that creating the consensus matrix seems to scale

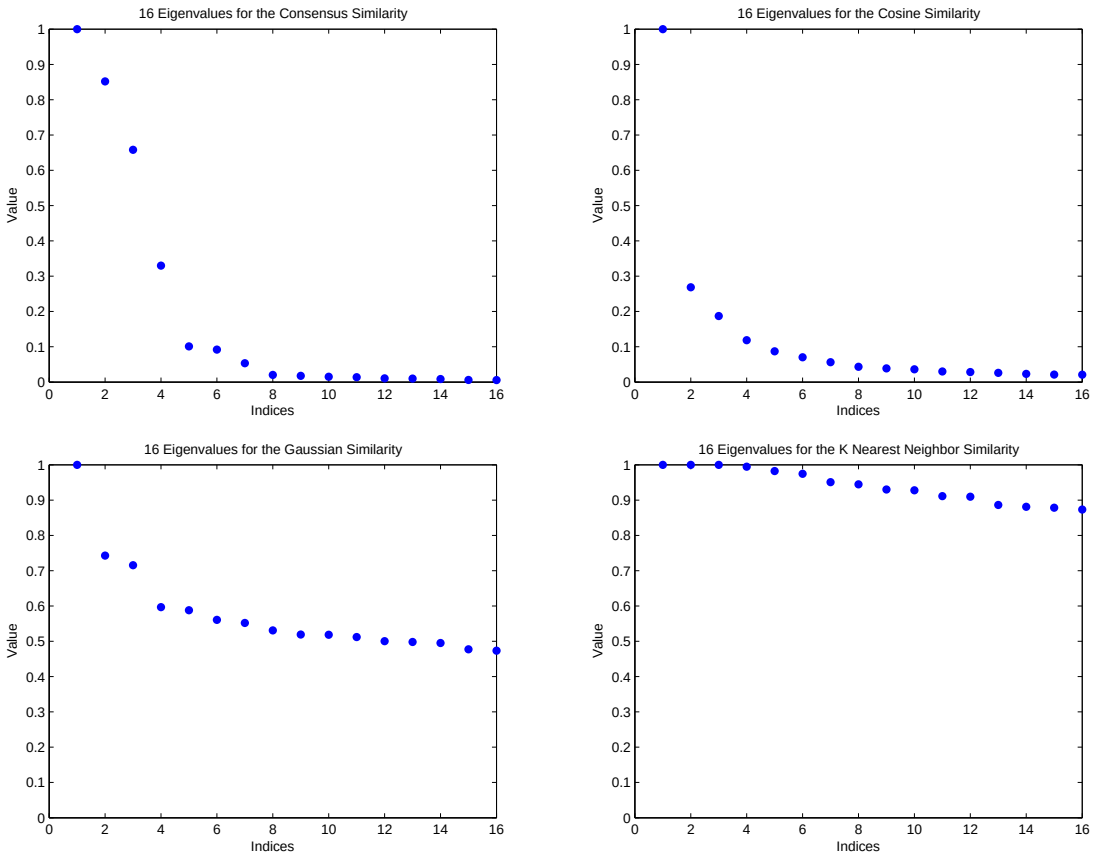

Figure 6.13: Eigenvalue plots for similarity matrices on the digits data set.

better with the number of observations. We discuss this issue in more detail at the end of this chapter.

Table 6.3: Accuracy and timing results for the digits data set.

	Gaussian Similarity Matrix	Consensus Similarity Matrix
Construction Time	633 sec	286 sec
Spectral Clustering Accuracy	0.7682	0.9162
Stochastic Clustering Accuracy	0.3532	0.9373

Figure 6.14 shows the visualization of the clustering as determined by the stochastic con-

sensus clustering algorithm. We have kept the same marker size as the previous examples, in order to make the plots easier to see. However, the number of observations has reached the point where a better resolution on the plot is desired. In practice we use a smaller dot size to better aid the user. We again also show the plot ordered by cluster membership to confirm that bands of probability values do capture the true clusters.

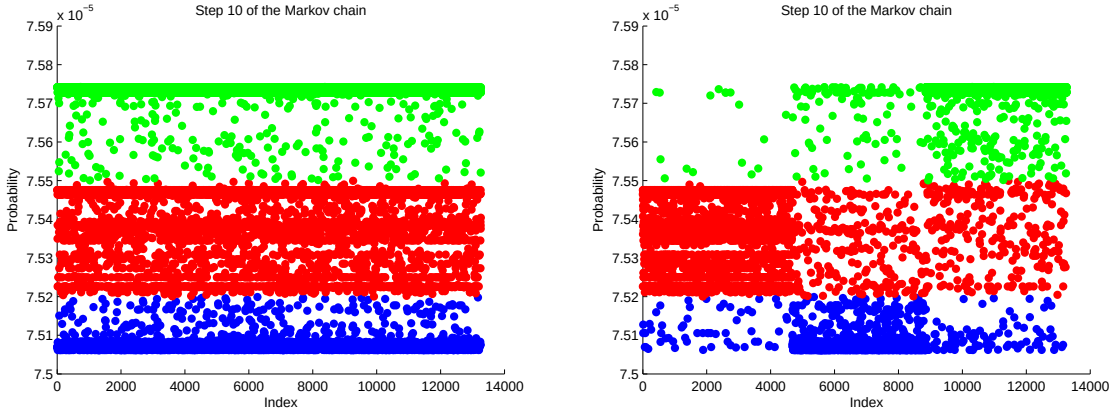

Figure 6.14: Visualization of the clustered data using the consensus similarity matrix on the digits data set. Indices are unordered (left) and ordered by true cluster (right).

We can compare the visualization of the clustering in Figure 6.14 to visualization of the data set using (PCA) in Figure 6.15. In the visualization using PCA it is not easy to see separation between the three clusters.

6.1.4 Small Practice Data Sets

We also include a table of the stochastic consensus clustering algorithm accuracy on several small practice data sets in Table 6.4. We do not go into great detail, as these data sets are small examples, but they do indicate that stochastic consensus clustering works on more than just the three examples we have discussed so far.

6.2 Exploring Data Sets Using Visualization

The next data sets that we use are ones in which we are not attempting to obtain the specific clusters that we know exist. These data sets represent scenarios that are closer to those experienced by data analysts in the world. The goal may be a predictive task, in which clustering

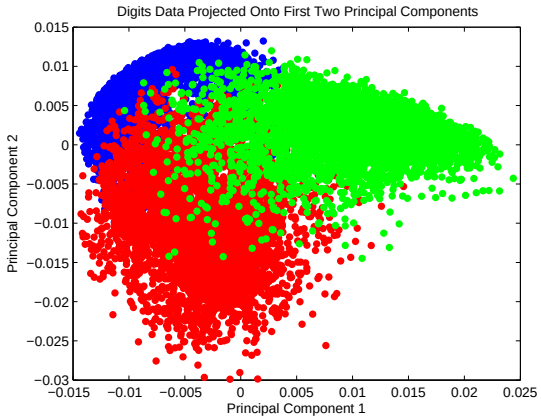

Figure 6.15: Visualization of the raw Digits data using PCA. The observations have been colored by cluster membership.

Table 6.4: Accuracy of the stochastic consensus clustering algorithm on small practice data sets.

	Iris data set [27]	Leukemia data set [24]	Ruspini data set [68]
accuracy	0.9667	0.9474	1.000

is used to improve the quality of the predictive models, or the goal may be to gain insight into the data by using clustering to explore the connections between various observations.

The use of clustering algorithms in this way is not new to stochastic consensus clustering. Rather, the benefit of the stochastic consensus clustering is the ability to visualize the steps of the Markov chain. A user can stop the Markov chain when data appears well-separated, and can create clusters based upon the perceived separation. Because of the Simon-Ando theory, we know this translates into a meaningful clustering result. We show this idea in action on the next few data sets.

As a side note, we use a smaller marker size for the upcoming examples than for the previous examples.

6.2.1 Spoken Letters Data Set [25]

This data set is referred to as the "Isolet data set" on the UCI repository [7]. This data set was formed by taking 150 human subjects and recording each person saying the letters of the alphabet twice. Each spoken letter of the alphabet is an observation, for a total of 7800

observations. The data separates 120 of the subjects into a training set and 30 subjects into a test set. In addition, a few observations have been dropped, yielding 6238 training observations and 1561 test observations.

We use the 6238 training observations, which have 617 dimensions. The stated goal for this data is to correctly determine what letter is spoken for a given observation. While we know what letter each observation corresponds to, we are not attempting to use clustering to classify each observation.

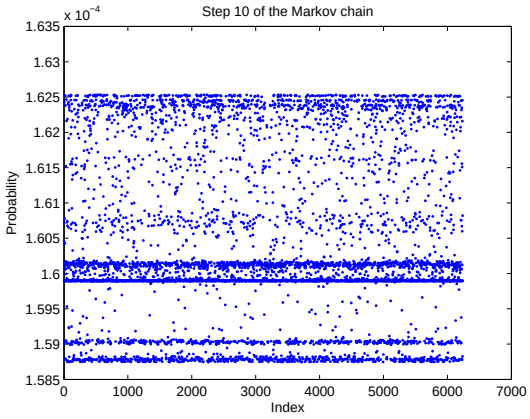

Figure 6.16: Visualization of the data using the consensus similarity matrix on the letters data set.

As discussed in Chapter 4, a user can observe each step of the Markov chain when using stochastic consensus clustering. For example, step 10 of the Markov chain for the letters data set is presented in Figure 6.16. We note that it appears that there are five dense bands of probability: two around 1.59×10^{-4}, two around 1.6×10^{-4}, and around 1.625×10^{-4}. There is one large non-dense area of probability from 1.605×10^{-4} to 1.62×10^{-4}.

From the discussion of the Simon-Ando theory we know that observations in the same cluster should all have around the same probability. Because of this, we assign each dense band of probability to a separate cluster, and the non-dense area as well to its own cluster. The resulting picture of this clustering can be seen in Figure 6.17.

We inspect the assignment of the clusters in Table 6.5. A letter has membership in a particular cluster if a notable majority of the observations for that letter are members of the cluster. In the case of "I" "L" and "W" the cluster membership was split. It is clear that many letters are clustered together due to phonological similarity.

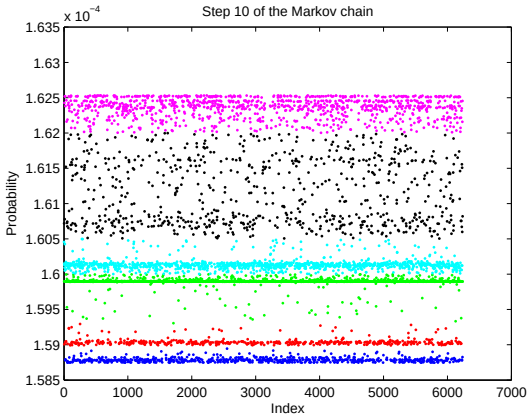

Figure 6.17: Visualization of the manually clustered data using the consensus similarity matrix on the letters data set.

Table 6.5: Letters comprising each cluster

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Letters	F S X	Q U W	B C D E G P T V Z	A H J K	I L M N W	I L O R Y

The separation of letters into distinct clusters allows for the creation of a predictive model by cluster. A model needs be more precise if the goal is to distinguish a "v" from a "c" than if the goal is to distinguish an "f" from a "w." While a human can easily separate letters by sound, this shows the capability of the stochastic consensus clustering to do the same on unknown data. Human interaction is still required, but only at the point of choosing the clusters from the visualization.

6.2.2 Steel Faults Data Set as found on the UCI repository[7]

Each observation in this data set represents a surface defect in a stainless steel plate. There are 27 attributes for each observation, and a total of 1941 observations in the data set. The original goal for this data set was to classify each observation one of six fault types, though there is a seventh group of "other faults."

We see iterations 3 and 4 of the Markov chain in Figure 6.18.

Upon manual inspection of the clusters we note that the stochastic clustering algorithm has separated the observations, not by fault type, but rather primarily by steel type, and secondarily by a variable titled: Outside_Global_Index.

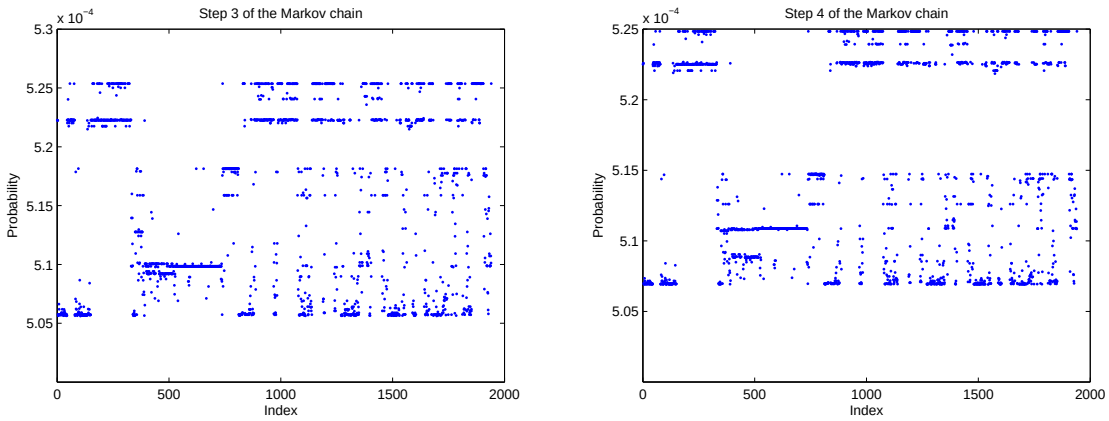

Figure 6.18: Visualization of index by probability plots using stochastic consensus clustering on the steel faults data set, ordered by steel fault type.

While the stochastic consensus clustering algorithm does not group each fault type into a distinct cluster, the algorithm does provide well separated clusters. This allows a user to create a more refined classification model, by creating a unique model for each cluster, as opposed to one model for the entire data set.

6.2.3 Million Song Data Set [10]

The data set we use is a subset of the Million Song Data Set that has been provided on the UCI data mining repository [7]. This is a subset both in the number of songs as well as in the number of attributes. This specific subset of the Million Song Data Set is intended for creating models to predict the year of the song.

- 515345 songs; we use a random 224801 song subset.
- 90 attributes: 12 for timbre average and 78 for timbre covariance.

Several initializations for the MSD data set resulted in probability plots that do not have noticeable bands of probability values, as all our previous examples do. The initialization that had the best plot is shown in Figure 6.19. We show steps four through seven of the Markov chain, and there appear to be two small areas of probability with one large blob of probability values in the center.

The interior blob of probability values is very wide, which is not the expected banded structure. We have several options for exploring the data more thoroughly: we can extract the

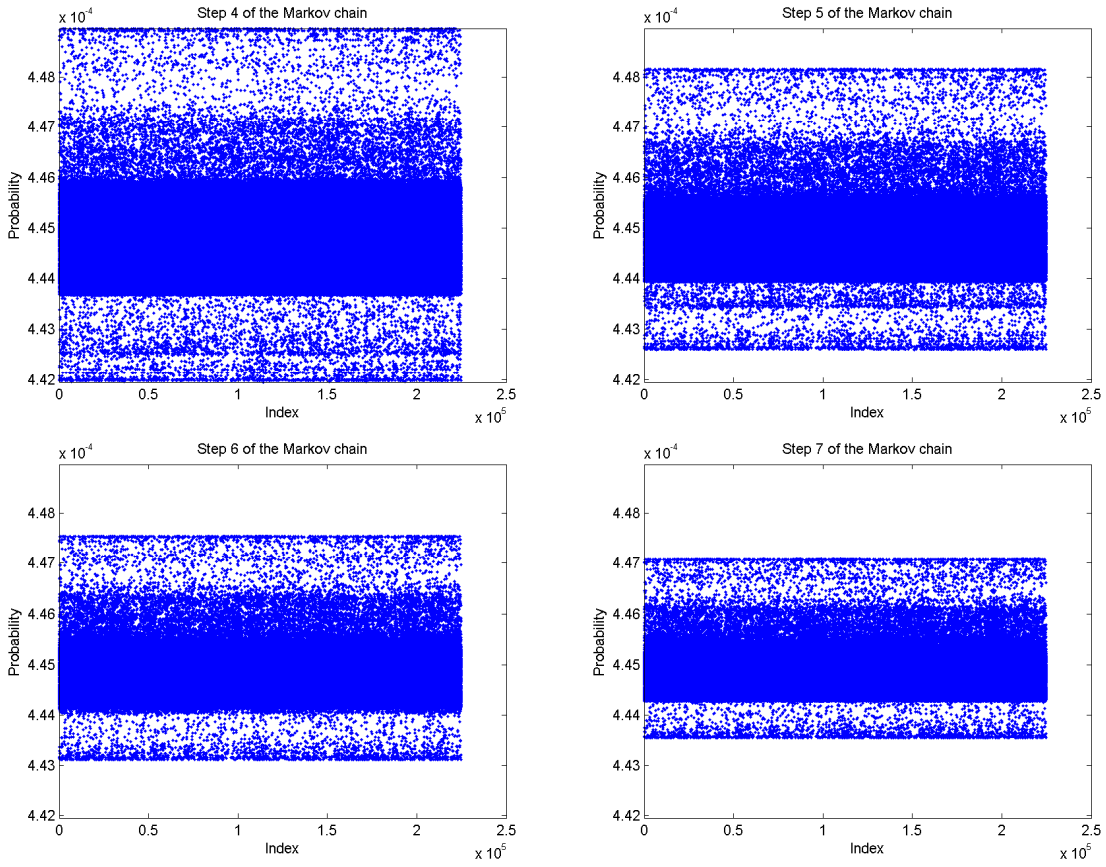

Figure 6.19: Visualization of probability by index plot, using stochastic consensus clustering on the Million Song Data Set. We keep the y axis the same in each plot to illustrate the converging Markov chain.

observations corresponding to the probability values in the middle and try to cluster this subset, or we can examine two simultaneous initializations of the Markov chain.

In Figure 6.20 we show four iterations of two separate initializations plotted against each other. If we look at each plot in order, we can see a movement of probability values that is unclear when considering each probability plot on its own.

In the middle of the plot of "Step 4 of the Markov chain" there is a group of probabilities that appear as a tail. In the subsequent plots, we see that this tail "moves" to the lower left portion of the plots, and develops a little hook. This movement of probabilities is more noticeable if we look at the pictures as an animated gif (from the early 90's internet days); however, we cannot show this on paper.

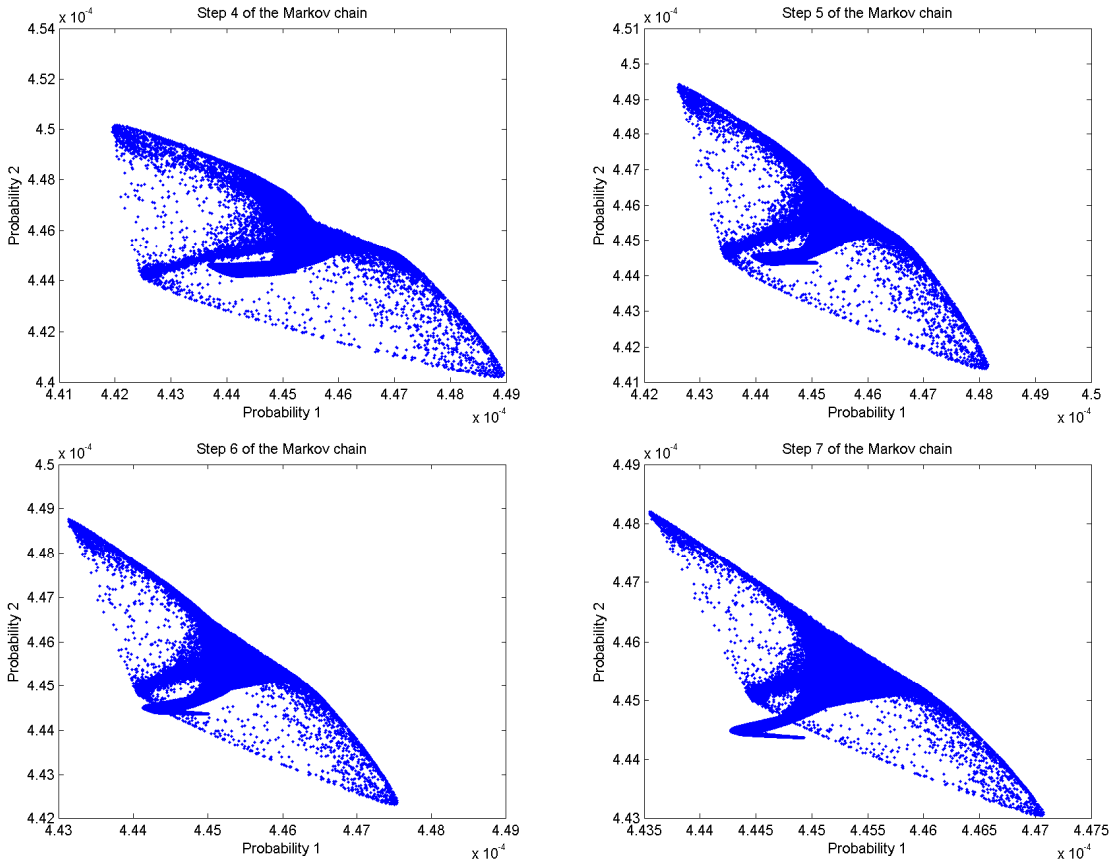

Figure 6.20: Visualization of probability by probability plots, using stochastic consensus clustering on the Million Song Data Set. The x and y axis are rescaled for each plot so the points are more noticeably distinct.

The plots in Figure 6.20 reveal that extra information is contained in the motion of the probability values from one step of the Markov chain to the next. In plotting only one initialization, as in Figure 6.19, we miss this motion because the motion takes place in the central blob of probability values.

Unlike the plots generated from other data sets, the probability value plots for the MSD do not indicate clusters as conclusively. We have several possible explanations:

- There was no underlying consensus among clustering for the consensus similarity matrix, or the clustering algorithms were not appropriate for this problem.
- There are too many observations for the Simon-Ando theory to be applied effectively. While Simon-Ando theory guarantees that observations in the same block will have an

approximately equal probability, it does not guarantee that other observations will not also have this probability.

- There are too many observations to visualize. While we cannot see the fine detail, meaningful blocks of density really do exist in the probability values.

We are of the opinion that the first and second explanations are the most likely. In the case of the first explanation, it is possible that there are several true core clusters and many observations that do not belong to any cluster more than others. Observations that don't belong to clusters will be split between clusters during the consensus matrix creation. Further research is needed on large data sets with known clusters to really determine what the issues may be. Unfortunately large data sets with well known clusters (that are not synthetically generated) are hard to find.

CHAPTER 7

Discussion and Concluding Remarks

We had three main goals in this thesis: to show the stochastic clustering algorithm generally performs well; to improve the original stochastic clustering algorithm; and to provide visualization to the user as a validation tool for the stochastic consensus clustering.

From the experiments in Chapter 6 we see that the stochastic clustering algorithm accurately assigns clusters on several known data sets when using the consensus similarity matrix. The cluster accuracy is high in each of the example data sets that we used, and the stochastic clustering algorithm outperformed spectral clustering each time. We showed the general performance of the stochastic clustering algorithm by ensuring that the data sets had a variety of sizes, as well as different application areas.

To improve on the stochastic clustering algorithm we addressed three future research questions posed in the original stochastic clustering paper. We experimentally showed that among the popular similarity matrices in the literature, the consensus similarity matrix was best suited for use in the stochastic consensus clustering. We replaced the Sinkhorn-Knopp algorithm with a new simultaneous scaling algorithm for balancing the consensus similarity matrix. The number of iterations required for the Sinkhorn-Knopp to converge is a magnitude larger than for the simultaneous scaling when run on the consensus similarity matrix. We showed this increase in speed experimentally in Chapter 6, in addition to showing the theoretical bound in the limit as discussed in Chapter 3. Finally, in Chapter 3 we improved the bounds that relate how uncoupled S and its doubly stochastic form P are.

We proposed a visualization method for the stochastic clustering algorithm that involves plotting the probability values for each iteration of the Markov chain. We provided examples

that show the bands of probability we expect from the Simon-Ando theory. We discussed how this visualization enables a user to understand if the clustering output is meaningful or not, and we extended this visualization to plot two initializations of the Markov chain at once.

The visualization of the Markov chain was especially useful in examples where we had no known cluster structure. In the case of the million song dataset, we were able to claim that the clustering result indicated clusters, although less definitively than the previous examples. The existence of the interesting results on the million song data set provides further research questions to pursue. If we had no visualization, then we would have been unable to know the quality of clustering output, and further questions would also be unavailable.

7.1 Conclusion

We summarize our contributions below:

7.1.1 Contributions

We introduced a measure $\mu(A)$ that we call the **row uncoupling measure of A**, and we showed that this measure is consistent with the theory of stochastic complementation, the underlying theory used in stochastic clustering. We related $\mu(A)$ to $\sigma(A, n_1)$, a measure given by Wessesl, in order to make use of his theoretical proofs on the nature of the eigenvalues of a nearly uncoupled matrix. Using our new measure, we were able to develop new bounds to show that when a nearly uncoupled matrix S is converted to a doubly stochastic matrix P, that P is also nearly uncoupled. These new bounds are important because $\mu(A)$ is more closely aligned to the underlying theory of stochastic complementation than the measure $\sigma(A, n_1)$.

We also introduced a new algorithm for balancing symmetric matrices into doubly stochastic form. We proved global convergence of this algorithm when $p = \frac{1}{2}$, and local convergence for all other $p \in (0, 1)$. We also showed that when $p \notin (0, 1)$ the algorithm will not converge. We continued by finding the rate of convergence of the algorithm in the limit using exact arithmetic. We examined the effect of finite precision by computing the absolute and relative error between double and quadruple precision calculations of the iterates of this algorithm. Finally, we showed that for consensus matrices, our new algorithm is more computationally efficient than the popular Sinkhorn-Knopp algorithm.

We introduced a visualization method for use with the stochastic clustering method. We claim that this visualization is not a replacement for standard visualization approaches, but is a computationally cheap way to help a user validate the results from the stochastic clustering.

We showed that stochastic clustering performs best on consensus matrices as opposed to several other popular similarity matrices. We also showed that the stochastic clustering performs

well on a wide variety of data sets of different sizes.

7.1.2 Future Research

We have identified several areas of future research and future goals from our results in this thesis.

- Investigate the relationship between the number of observations and the performance of the stochastic consensus clustering algorithm, with a focus on how our assumptions from Simon-Ando theory hold.
- Investigate the memory requirements of the consensus similarity matrix for large numbers of observations on distributed systems. Unlike other similarity matrices, this can be efficiently created in parallel.
- Investigate cluster performance and visualization in distributed systems.
- Investigate the use of a drop tolerance in relation to improving the consensus similarity matrix for clustering and enforcing a nearly uncoupled form [64].
- Seek an industry or topic-specific area in which stochastic consensus clustering adds value over currently used algorithms in the area.
- Investigate how to capture the "moving" probability values, as opposed to clustering specific iterations, or snapshots, of the Markov chain.
- Investigate improvements to the visualization method we have proposed, including a step in which we bin observations into quantiles and then plot only the mean of each quantile. The goal here would be to show the gaps by limiting the number of distinct points plotted on the graph.

REFERENCES

[1] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[2] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, 1999.

[3] D. F. Andrews. Plots of high-dimensional data. *Biometrics*, 28:125–136, 1972.

[4] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[5] Unknown Author. Visualizing principal components, 2012.

[6] Ana Azevedo and Manuel Filipe Santos. Kdd, semma, and crisp-dm: A parallel overview. In *IADIS European Conference in Data Mining*, pages 182–185, 2008.

[7] K. Bache and M. Lichman. Uci machine learning repository, 2013.

[8] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*, 2009.

[9] Abraham Berman and Robert J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Society for Industrial and Applied Mathematics, 1994.

[10] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[11] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer, 1981.

[12] Richard A. Brualdi and Bolian Liu. Fully indecomposable exponents of primitive matrices. *Proceedings of the American Mathematical Society*, 112(4):1193–2001, August 1991.

[13] Richard A. Brualdi, Seymour V. Parter, and Hans Schneider. The diagonal equivalence of a nonnegative matrix to a stochastic matrix. *Journal of Mathematical Analysis and Applications*, 16:31–50, 1966.

[14] Andreas Buja, Dianne Cook, and Deborah F. Swayne. Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics*, 5(1):78–99, 1996.

[15] Igor Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Visualization of navigation patterns on a web site using model-based clustering. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 280–284. ACM, 2000.

[16] Fan Chung. *Spectral Graph Theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. Conference Board of the Mathematical Sciences, 2007.

[17] Pierre Jacques Courtois. *Decomposability: Queueing and Computer System Applications*. Academic Press, 1977.

[18] J. Csima and B. N. Datta. The dad theorem for symmetric non-negative matrices. *Journal of Combinatorial Theory*, 1972.

[19] Ian Davidson. Visualizing clustering results. In *SIAM International Conference on Data Mining*, pages 3–18. Society for Industrial and Applied Mathematics, 2002.

[20] Jared L Dean. *Big Data, Data Mining, and Machine Learning*. Wiley, 2014.

[21] Gerard Debreu and I. N. Herstein. Nonnegative square matrices. *Econometrica*, 21(4):597–607, 1953.

[22] Inderjit Dhillon, Jacob Kogan, and Charles Nicholas. Feature selection and document clustering. In Michael W. Berry, editor, *Survey of Text Mining*, pages 73–100. Springer, 2004.

[23] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[24] T. R. Golub et al. Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[25] M. Fanty and R. Cole. Spoken letter recognition. In R. P. Lippman, J. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, chapter Spoken Letter Recognition. Morgan Kaufmann, 1991.

[26] Usama Fayyad, Georges G. Grinstein, and Andreas Wierse. *Information Visualization in Data Mining and Knowledge Discovery*. Morgan Kaufmann Publishers, 2002.

[27] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[28] E. W. Forgy. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, 21(3):768–769, 1965.

[29] Joel Franklin and Jens Lorenz. On the scaling of multidimensional matrices. *Linear Algebra and its Applications*, 114/115:717–735, 1989.

[30] Ping Fu. Personal correspondance, August 2014.

[31] Alexander N. Gorban and Andrei Yu. Zinovyev. Principal graphs and manifolds. *CoRR*, 2008.

[32] Peter Hall and Ker-Chau Li. On almost linearity of low dimensional projections from high dimensional data. *The Annals of Statistics*, 21(2):867 – 889, 1993.

[33] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistics Society. Series C (Applied Statistics)*, pages 100–108, 1979.

[34] Christopher Healy. Homepage. `http://www.csc.ncsu.edu/faculty/healey/`, 2014.

[35] Heiko Hoffmann. Kernel pca for novelty detection. *Pattern Recognition*, 40:863–874, 2007.

[36] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[37] R. A. Jarvis and Edward A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, C-22(11):1025–1034, November 1973.

[38] David J. Ketchen and Christopher L. Shook. The application of cluster analysis in strategic management research: An analysis and critique. *Strategic Management Journal*, 17(6):441–458, 1998.

[39] Phillip A. Knight. The sinkhorn-knopp algorithm: Convergence and applications. *SIAM J. Matrix Anal. Appl.*, 30(1):261–275, 2008.

[40] Phillip A. Knight and Daniel Ruiz. A fast algorithm for matrix balancing. *IMA Journal of Numerical Analysis*, 2012.

[41] Jacob Kogan. *Introduction to Clustering Large and High-Dimensional Data*. Cambridge University Press, 2007.

[42] Jrme Kunegis, Andreas Lommatzsh, and Christian Bauckhage. Alternative similarity functions for graph kernels. In *19th International Conference on Pattern Recognition*, pages 1–4, 2008.

[43] Serge Lang. *Undergraduate Analysis*. Springer, second edition, 1983.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[45] M. Lewin. On nonnegative matrices. *Pacific Journal of Mathematics*, 36(3):753–759, 1971.

[46] R. Lleti, M. C. Ortiz, L. A. Sarabia, and M. S. Snchez. Selecting variables for k-means cluster analysis by using a genetic algorithm that optimises the shilhouettes. *Analytica Chimica Acta*, 515(1):87–100, 2004.

[47] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[48] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[49] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. *Theoretical Computer Science*, 442:13–21, 2012.

[50] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1, 18, 1990.

[51] Christopher D. Manning, Prabhakar Raghavan, and Hinich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[52] Thiago G. Martins. Computing and visualizing pca in r, 2013.

[53] Carl D. Meyer. Stochastic complementation, uncoupling markov chains, and the theory of nearly reducible systems. *SIAM Review*, 31(2):240–272, 1989.

[54] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. Society of Industrial and Applied Mathematics, 2000.

[55] Henryk Minc. *Nonnegative Matrices*. John Wiley and Sons, 1988.

[56] Bojan Mohar. The laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, 2:871–898, 1991.

[57] Bojan Mohar. Some applications of laplace eigenvalues of graphs. In *Graph Symmetry: Algebraic Methods and Applications*, volume 497 of *NATO ASI Series*, pages 225–275. Springer, 1997.

[58] Alissar Nasser, Denis Hamad, and Chaiban Nasr. Kernel pca as a visualization tools for clusters identifications. *Artificial Neural Networks - ICANN 2006*, 4132:321–329, 2006.

[59] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, 2001.

[60] University of Utah. Scientific computing and imaging institute. `http://www.sci.utah.edu/`, 2014.

[61] B. N. Parlett and T. L. Landis. Methods for scaling to double stochastic form. *Linear Algebra and its Applications*, 48:53–79, 1982.

[62] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Springer, 2nd edition, 2007.

[63] Shaina Race. *Iterative Consensus Clustering*. PhD thesis, North Carolina State University, 2014.

[64] Shaina Race, Carl Meyer, and Keven Valakuzhy. Determining the number of clusters via iterative consensus clustering. In *Proceedings of SDM 2013*. SIAM, 2013.

[65] Radoop. Radoop big data analytics made easy. `http://www.radoop.eu/`, 2014.

[66] Matt Rasmussen and George Karypis. gcluto - an interactive clustering, visualization, and analysis system. Technical Report TR-04-021, University of Minnesota - Computer Science, 2004.

[67] Daniel Ruiz. A scaling algorithm to equilibrate both rows and columns noms in matrices. Technical Report RAL-TR-2001-034 and RT/APO/01/4, Rutherford Appelton Laboratory, 2001.

[68] E. H. Ruspini. Numerical methods for fuzzy clustering. *Information Science*, 2:319–350, 1970.

[69] Nikos A. Salingaros. Complexity and urban coherence. *Journal of Urban Design*, 5(3):291–316, 2000.

[70] SAS. Sas enterprise miner homepage. `http://www.sas.com/en_us/software/analytics/enterprise-miner.html`, 2014.

[71] Aaron Schumacher. Pca, 3d visualization, and clustering in r, 2013.

[72] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[73] Herbert A. Simon. Near decomposability and the speed of evolution. *Industrial and Corporate Change*, 11(3):587–599, 2002.

[74] Herbert A. Simon and Albert Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29(2):111–138, 1961.

[75] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[76] George W. Soules. The rate of convergence of sinkhorn balancing. *Linear Algebra and its Applications*, 150:3–40, 1991.

[77] O. Sporns, G. Tononi, and G. M. Edelman. Connectivity and complexity: the relationship between neuroanatomy and brain dynamics. *Neural Networks*, 13(8-9):909–922, 2000.

[78] Statsoft. *Electronic Statistics Textbook*, chapter Finding the Right Number of Clusters in k-Means and EM Clustering: v-Fold Cross-Validation. Statsoft Inc., 2013.

[79] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc., 2005.

[80] Georgia Tech. `http://fodava.gatech.edu/fodava-testbed-software`, 2014.

[81] Robert L. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953.

[82] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clustering in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B*, 63:411–423, 2001.

[83] Belle L. Tseng, Junichi Tatemura, and Yi Wu. Tomographic clustering to visualize blog communities as mountain views. In *In WWW 2005 Workshop on the Weblogging Ecosystem*, 2005.

[84] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[85] R. von Mises and H. Pollaczek-Geiringer. Praktische verfahren der gelichungsauflsung. *ZAMM - Zeitschrift fr Angewandte Mathematik und Mechanik*, 9:152–164, 1929.

[86] Charles Wessell. *Stochastic Data Clustering*. PhD thesis, North Carolina State University, 2011.

[87] Charles Wessell. Stochastic data clustering. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1214–1236, 2012.

[88] Matthew O. Ward Ying-Huey Fua and Elke A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *Visualization '99. Proceedings*, pages 43–50, 1999.

APPENDICES

APPENDIX A

Stochastic Clustering Matlab GUI

Throughout the thesis we argue that visualization is an important tool when paired with clustering. We claim that visualization is natural and easy in the process of running the stochastic clustering algorithm.

We have made a graphical user interface (GUI) for Matlab, which allows a user to load in a similarity matrix, and perform the stochastic clustering on the matrix, along with visualization. We outline a simple example using the MCC data. We will host the appropriate matlab files at meyer.math.ncsu.edu.

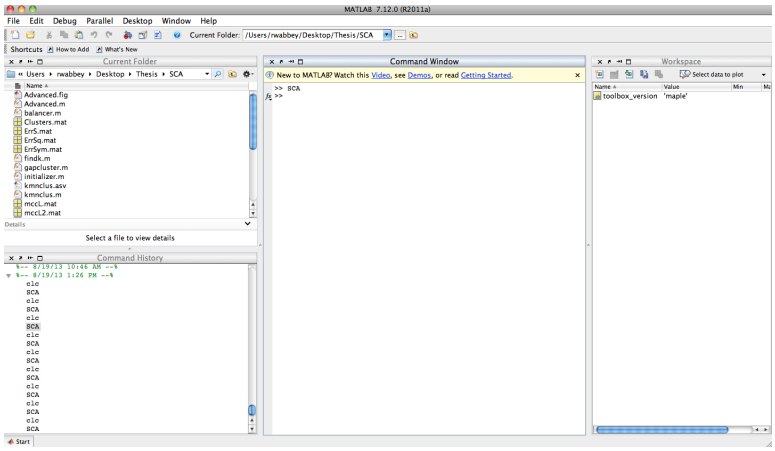

Figure A.1: First call the GUI using the command SCA in Matlab.

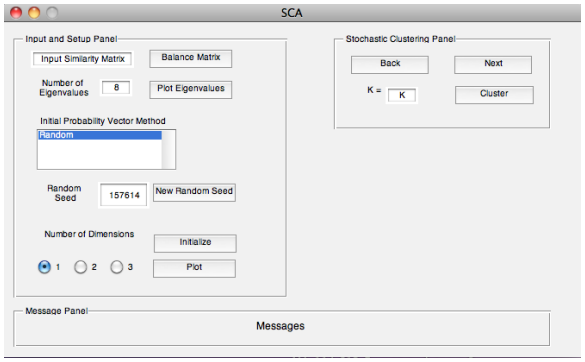

Figure A.2: This is the interface the user first sees.

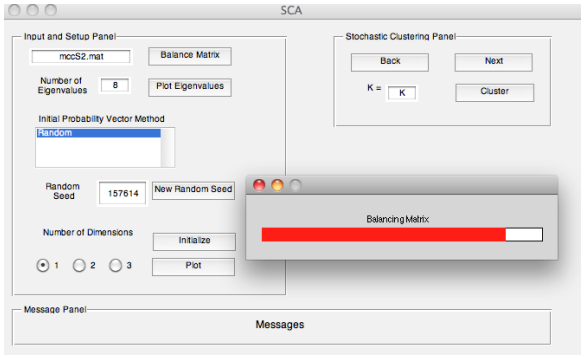

Figure A.3: Type in the .mat file, and click 'balance matrix'. Then click 'initialize' and then click 'plot'.

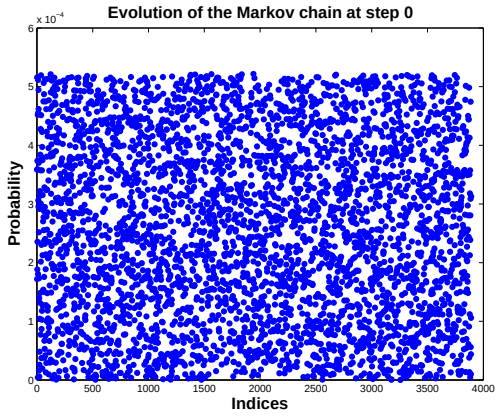

Figure A.4: The first plot that is created by the GUI.

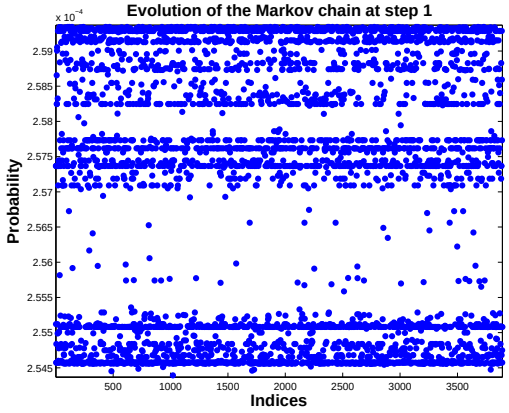

Figure A.5: By clicking 'next', we get this plot.

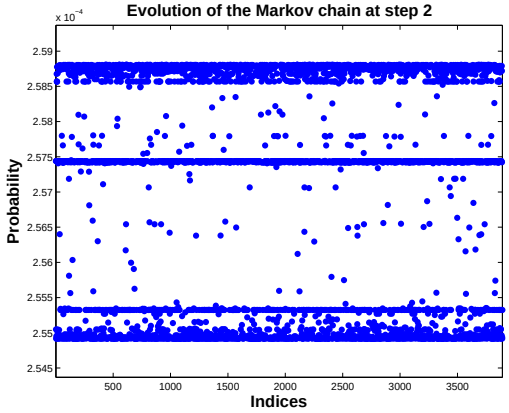

Figure A.6: By clicking 'next' again, we get this plot.

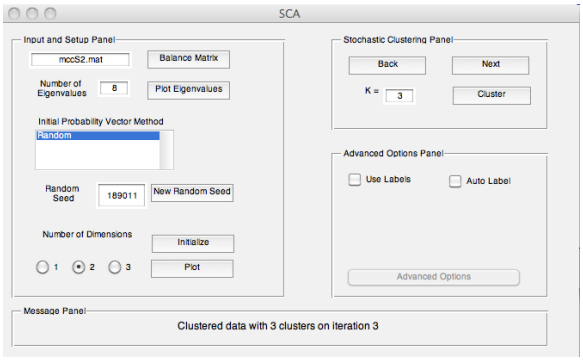

Figure A.7: Input 3 into 'k=' and then click 'cluster'.

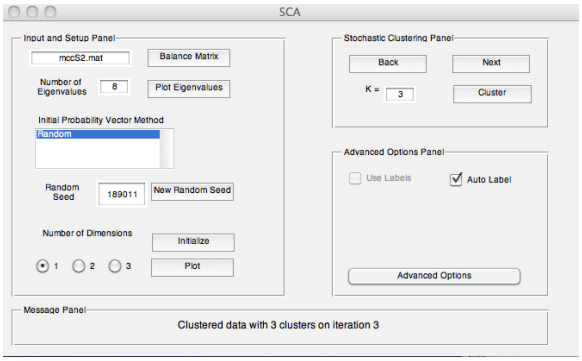

Figure A.8: Click 'auto label' to label the indices automatically.

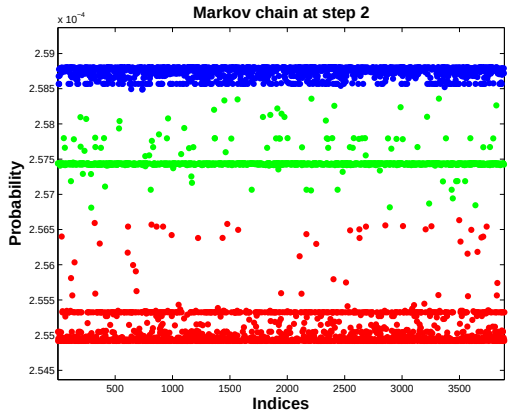

Figure A.9: The previous step 2 plot now clustered.

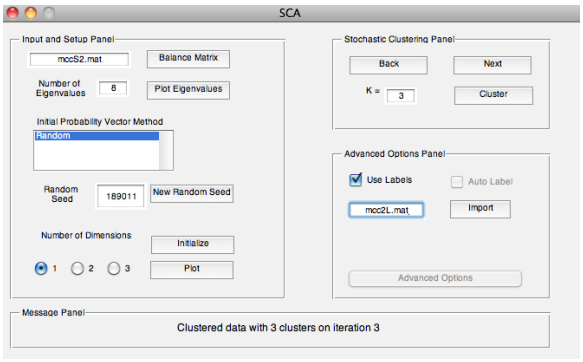

Figure A.10: Instead we want to import a known clustering. Click 'use labels' type in the .mat file name, and click 'import'.

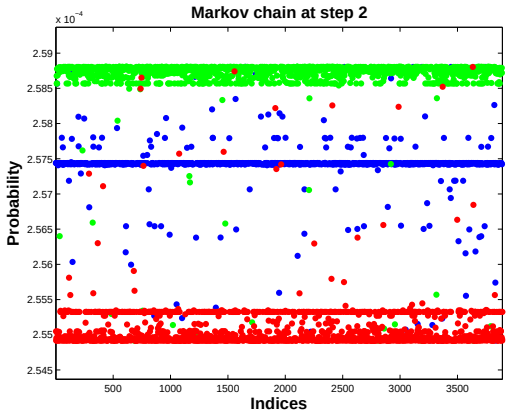

Figure A.11: Now the indices are colored according to true cluster membership.

APPENDIX B

Distributed Memory Stochastic Clustering

While not used for the experiments in this thesis, we felt that a discussion on distributed memory computing is important. Many large companies need to resort to distributed memory computing, as they acquire much more data than they can store on a single machine.

The stochastic clustering algorithm requires a similarity matrix as an input. By nature all similarity matrices are $n \times n$ matrices, as they describe the similarity of every observation to every other observation. A double precision number in C often requires 8 bytes of storage. With 48 gibibytes of storage (48×2^{30} bytes), the maximum n is around 80,000. This is by no means large data.

Often times drop tolerances are used in the case of similarity matrices to enforce some degree of sparsity. This serves two goals: that the data can now fit in memory, and, by dropping small similarities, clusters become more separated. However, drop tolerances can only help to a certain extent; at some point it becomes necessary to distribute the original data, and also the similarity matrix.

B.1 Distributed Similarity Construction

If the original data is distributed across multiple machines, it becomes time-consuming to construct large similarity matrices. As each observation needs to be compared to every other observation the data needs to be passed around from machine to machine, as each calculates pairwise comparisons of the various observations that the machines store in memory.

For the consensus similarity matrix this is different. The consensus matrix requires the

input from clustering algorithms. As such, if clustering methods are optimized to run in a distributed memory environment, then the clustering results will be quickly obtained. It is quite fast to construct the sparse vector format for a given clustering adjacency matrix. After several clustering algorithms have been run, these matrices only need be added together. If each machine in the distributed memory environment is responsible for specific rows of the consensus similarity matrix, then no passing of data is required after the clustering algorithms have been run and the clustering results shared with all machines.

B.2 Distributed Stochastic Clustering

Once a similarity matrix has been constructed, the stochastic clustering algorithm relies only on matrix vector products. This is true both for balancing the matrix, and for watching the Markov chain. The doubly stochastic matrix P should never be calculated, and instead only the vector $x_\star$. By using the simultaneous scaling algorithm we can be sure that if a cluster structure exists in the similarity matrix, the algorithm will have error less than 10^{-10} within 40 iterations (assume a decrease in error by 1/2 for each iteration).

Ultimately we feel confident that stochastic consensus clustering in a distributed environment will be much faster than spectral clustering. Spectral clustering requires the computation of the smallest k eigenvalues, eigenvector pairs. In a distributed memory environment this will take much more time than 60 matrix vector multiplications.