CLUSTERING IN RELATIONAL DATA AND ONTOLOGIES

A Dissertation presented to
the Faculty of the Graduate School
University of Missouri

In Partial Fulfillment
of the Requirements for the Degree
of Doctor of Philosophy

---

by

TIMOTHY C. HAVENS

Dr. James M. Keller, Dissertation Supervisor

July 2010

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

CLUSTERING IN RELATIONAL DATA AND ONTOLOGIES

presented by TIMOTHY C. HAVENS,

a candidate for the degree of Doctor of Philosophy,

and hereby certify that in their opinion it is worthy of acceptance.

Dr. James M. Keller

Dr. Marjorie Skubic

Dr. Mihail Popescu

Dr. Jack C. Schultz

Dr. Gregory L. Alexander

I give my most heart-felt thanks to my friend, editor, biggest critic, romantic colleague and partner, Steph — without her I would never have written this. I thank Steph for providing me with the motivation to move to Missouri and meet all the wonderful people I have worked with and befriended. Beyond this, she has been a source of strength for me in innumerable ways.

To Steph, I dedicate this dissertation.

ACKNOWLEDGMENTS

This dissertation would not have been possible without the contributions, help, and friendship of so many people. Here, I specifically mention a few.

In particular, I am grateful for the ceaseless support and love of my parents. Their unwavering faith and confidence in me and my abilities have shaped me into the person I am today.

I would like to thank my committee, Marjorie Skubic, Greg Alexander, Jack Schultz, and Mihail Popescu. I would like to thank Marge for introducing me to the emotionally-satisfying and pertinent field of eldercare. Thank you, Greg, for teaming up with me on the RAND grant and for your continuing encouragement and mentoring—I look forward to our future collaborations! Discovering that Jack and I shared a passion for both jazz and science was certainly a highlight, and I would like to thank you, Jack, for the great collaborations, on the stage and in the lab. Finally, I would to thank Mihai for his friendship, collegiality, and his shared love for clustering and relational data.

This dissertation topic would never have existed if Jim Bezdek had not visited the University of Missouri and sparked an explosion of interest in clustering. My outlook on science, writing, life, and cigars is forever changed, thanks to you, Jim. I look forward to our continued collaborations, friendship, and adventures.

Last, but certainly not least, I would like to thank my amazing advisor, Jim Keller. I appreciate your patience throughout this process and countless hours of reading, encouraging, and reflecting on this dissertation. Furthermore, I thank you for taking me around the world, introducing me the best and the brightest in our field, and for showing me that being a professor is the best job in the world. I will forever treasure my experiences at the University of Missouri, and you made that possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AFMS	Augmented Fuzzy Measure-based Similarity
BE	Bond Energy
BSAS	Basic Sequential Algorithmic Scheme
CCE	Cluster Count Extraction
CCV	Cluster Correlation Validity
CI	Computational Intelligence
CLODD	Clustering in Ordered Dissimilarity Data
CS	Compact-Separated
CWS	Compact-Well-Separated
DAG	Directed-Acyclic-Graph
FCM	Fuzzy c-Means
FI	Furthest Insertion
FMS	Fuzzy Measure-based Similarity
GO	Gene Ontology
LOS	Linear combination of Order Statistics
LPm	Linear Placement
MST	Minimal Spanning Tree
NCA	Nearest Common Ancestor
NERFcM	Non-Euclidean Relational Fuzzy c-Means
NI	Nearest Insertion
NN	Nearest Neighbor
OSOM	Ontological Self-Organizing Map
PCM	Possibilistic c-Means
PSO	Particle Swarm Optimization
ReSL	Rectangular Single Linkage
RDI	Reordered Dissimilarity Image
RGB	Red-Green-Blue
SL	Single-Linkage
SOM	Self-Organizing Map
SOSM	Self-Organizing Semantic Map
TSP	Traveling Salesman Problem
VAT	Visual Assessment of Cluster Tendency

ABSTRACT

This dissertation studies the problem of clustering objects represented by *relational data*. This is a pertinent problem as many real-world data sets can only be represented by relational data for which object-based clustering algorithms are not designed. Relational data are encountered in many fields including biology, management, industrial engineering, and social sciences. Unlike *numerical object data*, which are represented by a set of feature values (e.g. height, weight, shoe size) of an object, relational object data are the numerical values of *(dis)similarity* between objects. For this reason, conventional cluster analysis methods such as k-means and fuzzy c-means cannot be used directly with relational data.

I focus on three main problems of cluster analysis of relational data: (i) *tendency prior to clustering*—how many clusters are there?; (ii) *partitioning of objects*—which objects belong to which cluster?; and (iii) *validity of the resultant clusters*—are the partitions "good"? Analyses are included in this dissertation that prove that the *Visual Assessment of cluster Tendency* (VAT) algorithm has a direct relation to *single-linkage* hierarchical clustering and *Dunn's cluster validity index*. These analyses are important to the development of two novel clustering algorithms, **CLODD**-*CLustering in Ordered Dissimilarity Data* and **ReSL**-*Rectangular Single-Linkage* clustering.

Also presented in my analysis of VAT is a recursive formulation of the *improved* VAT (iVAT) algorithm. iVAT is shown to improve the visual evidence of cluster tendency on some types of data for which VAT fails. The computational complexity of my recursive formulation of iVAT is $O(n^2)$, as opposed to $O(n^3)$ of the original formulation—n is the number of objects considered.

CLODD is a clustering algorithm that works on reordered dissimilarity data. Typically, the dissimilarity matrix is reordered with the VAT algorithm; although, I

present results on data that is reordered with other methods. CLODD produces partitions by minimizing a novel objective function that measures the fit of a candidate partition to the 'blockiness' of images of reordered dissimilarity data. Three characteristics are included in the objective function: 'blockiness' or constrast, 'edginess', and small-cluster rejection. CLODD is shown to extract good partitions of data when the reordering method is successful in producing "good" visualizations.

A special form of relational data is rectangular data. Rectangular relational data are dissimilarity values between a set of row objects and a set of column objects, where the relation among row objects (or column objects) is unknown. One example of this type of data set is ratings given by movie reviewers (the column objects) for a set of movies (the row objects). Each rating is interpreted as a dissimilarity value between a reviewer and a movie—high rating indicates low dissimilarity, while a low rating indicates high dissimilarity. Additionally the relation between reviewers (or between movies) is unknown.

I present a new formulation of the co-VAT algorithm, which a visual method for determining the cluster tendency of rectangular data. I provide several examples for which my new formulation is successful in showing the preferred cluster tendency and for which the original co-VAT fails. I also extend the notions in the iVAT algorithm to the co-VAT algorithm, which I call **co-iVAT**.

ReSL addresses a special form of relational data, rectangular data. ReSL extracts five types of clusters from rectangular relational data. I present several examples that show the behavior of ReSL in the presence of different types of rectangular data. A comparison is made with spectral co-clustering and ReSL is shown to more effective at elucidating the clustering structure.

Last, this dissertation addresses clustering in ontologies, a specific type of data. These data are composed of collections of terms organized in a hierarchical tax-

onomy (a directed acyclic graph); examples include the Gene Ontology, the MeSH ontology, patient medical records, and web documents. I apply an extension to the Self-Organizing Map (SOM) to produce a new algorithm, the **OSOM**-*Ontological Self-Organizing Map*. OSOM provides visualization and linguistic summarization of ontology-based data. A binary-valued vector-based network prototype is used to represent ontological objects (e.g. genes and gene products).

These algorithms and analyses are pertinent to all problems that produce relational data; however, I specifically examine these methods, especially the OSOM, for use on bioinformatics-based data. I show results of these algorithms with data composed of Gene Ontology annotations and microarray gene expression data, and with data available from the UCI Machine Learning Repository.

Chapter 1

Introduction

John McCarthy described *artificial intelligence* in 1956 as “the science and engineering of making intelligent machines.”[Skillings, 2006] In the proposal for the conference at which he coined this term McCarthy stated, “the major obstacle is not the lack of machine capacity but our inability to write programs taking full advantage of what we have.”[McCarthy et al., 1955] We continue to struggle with this aspect of building intelligent machines. This dissertation contributes to the field of artificial intelligence by describing the development of a novel set of algorithms that perform a task that humans perform effectively and efficiently: sorting objects into groups according to context.

Since the 1956 Dartmouth Summer Research Project on Artificial Intelligence, at which McCarthy created the term *artificial intelligence*, computer programs have become the tool of choice for the development of artificial intelligence and subjects such as neural networks, fuzzy systems, swarm intelligence, and evolutionary computation have become popular research areas. These subjects are often lumped together into the field of *computational intelligence* (CI). Noteworthy general references on CI are the works by Engelbrecht [2007] and Pedrycz [1997]. Poole et al. [1998] defined CI as "the study of the design of intelligence agents," describing an intelligent agent as a "system that acts intelligently." Table 1.1 contains the definitions of *intelligence* from Merriam-Webster [Mer, 2009]. I argue that definition 5. is not sufficient to define an intelligence agent as my calculator watch in grade school had the ability to "perform computer functions", but it could not act intelligently. As the most applicable to CI, let us examine, in particular, definition 1.a.

Table 1.1: Definitions of *intelligence* [Mer, 2009].

1. **a** (1): the ability to learn or understand or to deal with new or trying situations; also: the skilled use of reason, (2): the ability to apply knowledge to manipulate one's environment or to think abstractly as measured by objective criteria (as tests); **b**: Christian Science: the basic eternal quality of divine Mind; **c**: mental acuteness

2. **a**: an intelligent entity; **b**: intelligent minds or mind <cosmic intelligence>

3. the act of understanding

4. **a**: information, news; **b**: information concerning an enemy or possible enemy or an area; also: an agency engaged in obtaining such information

5. the ability to perform computer functions

This definition states that intelligence is "the ability to learn or understand or to deal with new or trying situations" and "the ability to apply knowledge to manipulate one's environment or to think abstractly as measured by objective criteria." Hence, an intelligent agent must have these abilities. While it could be argued that artificial systems exist that display these traits, I believe that humans are far from creating a truly intelligent system (I should mention that famed futurist, Raymond Kurzweil, predicts that a computer will pass the *Turing Test* [Turing, 1950] in 2029 [Kurzweil, 2006]). However, we have come far since 1956: in that same year Ulam's MANIAC I became the first chess program to defeat a human; in 1959 a checkers program won against the best human player in the world; in 1962 the first commercial industrial robots were born; in 1968 the (fictitious) artificial intelligence HAL was a leading character in the film 2001: A Space Odyssey; in 1986 a chess program competed at the senior master level; in 1989 Deep Thought defeated International Chess Master David Levy; in May 1997 Deep Blue defeated chess world champion Garry Kasparov; and currently millions of people around the world are playing video games with extremely complex artificial intelligence components. To add additional perspective to this timeline, on Nov. 25, 2006, chess world champion Vladimir Kramnik was defeated by a commercially available chess program, Deep Fritz [McClain, 2006]. Professor Monty Newborn of McGill University summed up the event, "I don't know what one could get out of it [computer chess] at this point. The science is done."

CI has certainly progressed rapidly; however, the science is far from done. Two areas of human intelligence that CI has not mastered are classification and clustering. Humans naturally classify or group and, subsequently, name objects, situations, and abstractions they encounter in everyday life. This ability is vitally important as it gives humans the power to address new situations by comparing the traits of new objects or phenomena with known traits; thus, a human can act intelligently even in "new and trying situations" (see definition of *intelligence* in Table 1.1). Clustering is the act of sorting unlabeled (or uncategorized) objects into groups according to their traits; often the number of groups or the reason that certain objects are grouped together is unknown (at least, before the clustering process is completed). The traits of objects in the same group should be similar to one another, and the object traits in different groups should be dissimilar from one another [Jain et al., 1999]. This is in contrast to the classical classification problem in which objects are labeled as belonging to one (or many) predefined categories.

Clustering algorithms are an integral part of both CI and pattern recognition.

Often researchers are mired in data sets that are large and unlabeled. There are many methods by which researchers can elucidate these data, including projection and statistical methods. Clustering provides another tool for deducing the nature of the data by providing labels that describe how the data separates into groups. Clustering has also been shown to improve the performance of other algorithms or systems by separating the problem-domain into manageable sub-groups—a different algorithm or system is tuned to each cluster [Frigui, 2007, Bo and Nevatia, 2007]. Clustering has also been used to infer the properties of unlabeled objects by clustering these objects together with a set of labeled objects (of which the properties are well understood) [Khan et al., 2003, The UniProt Consotium, 2007].

The problem domains and applications of clustering are innumerable. Virtually every field, including biology, engineering, medicine, finance, mathematics, and the arts, have used clustering. Its function—grouping objects according to context—is a basic part of intelligence and is ubiquitous to the scientific endeavor. This dissertation will examine a specific, but general, form of clustering: clustering in relational data.

1.1 The Problem

Consider a set of objects $\mathbf{O} = \{o_1, \ldots, o_n\}$. These objects can represent virtually anything—vintage bass guitars, pure-bred cats, cancer genes expressed in a microarray experiment, cake recipes, or web-pages. The object set $\mathbf{O}$ is *unlabeled data*; that is, each object has no associated class label. However, it is assumed that there are subsets of similar objects in $\mathbf{O}$. These subsets are called *clusters*.

Numerical object data is represented as $\mathbf{X} = \{\vec{x}_1, \ldots, \vec{x}_n\} \subset \mathbb{R}^p$, where each dimension of the vector $\vec{x}_i$ is a feature value of the associated object o_i. These features can be a veritable cornucopia of numerical descriptions, i.e., RGB values, gene expression, year of manufacture, number of stripes, etcetera. Another way to represent the objects in $\mathbf{O}$ is with numerical *relational* data, which consist of n^2 values that represent the (dis)similarity between pairs of objects. These data are commonly represented by a relational matrix $\mathbf{R} = [r_{ij} = \text{relation}(o_i, o_j) | 1 \leq i, j \leq n]$. The relational matrix can take one of two forms, a *similarity* matrix $\mathbf{S}$ or a *dissimilarity* matrix $\mathbf{D}$. Similarity or dissimilarity values are interchangeable with several well-known transformations, the most famous being $\mathbf{D} = [1] - \mathbf{S}$ for $\mathbf{D} \in [0, 1]$. Dissimilarity can usually be interpreted as a distance between objects. For instance, numerical data $\mathbf{X}$ can always be converted to $\mathbf{D}$ by $d_{ij} = \|\vec{x}_i - \vec{x}_j\|$ (any vector norm on $\mathbb{R}^p$). As a result, relational algorithms can be used with both numerical object data $\mathbf{X}$ and relational data $\mathbf{R}$, viz. $\mathbf{D}$ or $\mathbf{S}$. There are, however, similarity and dissimilarity relational data sets that do not begin as numerical object data; for these, there is no choice but to use a relational algorithm. Hence, relational data represent the "most general" form of input data. This dissertation will pose most problems in terms of the dissimilarity $\mathbf{D}$; however, there are some situations where discussion of similarity $\mathbf{S}$ is more intuitive.

Another form of relational data is rectangular. These data are represented by an $m \times n$ dissimilarity matrix $\mathbf{D}$, where the entries are the pair-wise dissimilarity values between m row objects $\mathbf{O}_r$ and n column objects $\mathbf{O}_c$. An example of a rectangular relational data set is a movie review database, where row objects are m movies and column objects are n movie reviewers. A real-world example is the Netflix Prize data (http://www.netflixprize.com). Another example comes from web-document analysis, where the row objects are m web-pages, the columns are n words, and the (dis)similarity entries are occurrence measures of words in web-pages [Kummamuru et al., 2003, Dhillon et al., 2003, Dhillon, 2001]. In each case, the row and column objects are non-intersecting sets, such that the pair-wise relation among row (or column) objects is unknown. Conventional relational clustering algorithms are ill-equipped to deal with rectangular data. Additionally, the definition of a cluster as a group of similar objects takes on a new meaning. There can be groups of similar objects that are composed of only row objects (movies that get good reviews), of only column objects (reviewers that like the same movies), or of mixed objects (the movies that received good reviews and the reviewers that gave them). In all, there are five types of clusters in rectangular data. Rectangular clustering, often called *co-clustering* or *bi-clustering*, is described in more detail in Section 2.3 and in references, [VanMechelen et al., 2004] and [Kriegel et al., 2009].

A wide array of algorithms exists for clustering unlabeled object data $\mathbf{O}$. Descriptions of many of these algorithms, both relational and not, can be found in the following general references on clustering: Duda et al. [2000], Theodoridis and Koutroumbas [2009], Bezdek [1981], Bezdek et al. [1999], Hartigan [1975], Xu and Wunsch II [2009], Jain et al. [1999], and Jain and Dubes [1988]. *Clustering* in unlabeled data $\mathbf{X}$ or $\mathbf{D}$ is defined as the assignment of *labels* to groups of similar (unlabeled) objects $\mathbf{O}$. In other words, objects are *sorted* or *partitioned* into groups such that each group is composed of objects with similar traits. There are two important factors that all clustering algorithms must consider: 1) the number (and, perhaps, type) of clusters to seek and, 2) a mathematical way to determine the similarity between various objects (or groups of objects). Let c denote the integer number of clusters. The number of clusters can take the values $c = 1, 2, \ldots, n$, where $c = 1$ results in the universal cluster (every object is in one cluster) and $c = n$ results in single-object clusters.

A *partition* of the objects is defined as the set of cn values, where each value $\{u_{ik}\}$ represents the degree to which an object o_k is in (or represented by) the ith cluster. The c-partition is often arrayed as a $c \times n$ matrix $\mathbf{U} = [u_{ik}]$, where each row represents a cluster and each column represents an object. There are three types of partitions (to date), crisp, fuzzy (or probabilistic), and possibilistic [Bezdek, 1981, Krishnapuram and Keller, 1993]. *Crisp* partitions of the unlabeled objects are non-empty mutually-disjoint subsets of $\mathbf{O}$ such that the union of the subsets cover $\mathbf{O}$. The set of all non-degenerate (no zero rows) crisp c-partition matrices for the object set $\mathbf{O}$ is:

$$\mathbf{M}_{hcn} = \{\mathbf{U} \in \mathbb{R}^{cn} | u_{ik} \in \{0,1\}\, \forall i,k; \sum_{i=1}^{c} u_{ik} = 1\, \forall k; \sum_{k=1}^{n} u_{ik} > 0\, \forall i\}, \tag{1.1}$$

where u_{ik} is the *membership* of object o_k in cluster i; the partition element $u_{ik} = 1$ if o_k is labeled i and is 0 otherwise.

Fuzzy (or probabilistic) partitions are more flexible than crisp partitions in that each object can have membership in more than one cluster. Note, if $\mathbf{U}$ is probabilistic, the partition values are interpreted as a probability $p(i|o_k)$ that o_k is in the i-th class. This dissertation will assume that fuzzy and probabilistic partitions are essentially equivalent from the point of view of clustering algorithm development. The set of all fuzzy c-partitions is:

$$\mathbf{M}_{fcn} = \{\mathbf{U} \in \mathbb{R}^{cn} | 0 \le u_{ik} \le 1\, \forall i,k; \sum_{i=1}^{c} u_{ik} = 1\, \forall k; \sum_{k=1}^{n} u_{ik} > 0\, \forall i\}. \tag{1.2}$$

Each column of the fuzzy partition $\mathbf{U}$ must sum to 1, thus ensuring that every object is completely partitioned ($\sum_i u_{ik} = 1$).

Possibilistic partitions relax this condition, allowing partition columns that do not necessarily sum to 1. Possibilistic clustering has been shown to be especially effective in partitioning data that has outliers and intersecting clusters [Krishnapuram and Keller, 1993]. The set of all possibilistic c-partitions is:

$$\mathbf{M}_{pcn} = \{\mathbf{U} \in \mathbb{R}^{cn} | 0 \le u_{ik} \le 1\, \forall i,k; \forall k \exists i \ni u_{ik} > 0\}, \tag{1.3}$$

where u_{ik} is the possibility that o_k is in cluster i (this has also been described by Krishnapuram and Keller [1993] as the *typicality* of o_k to cluster i). Notice that the possibilistic partition ensures that there is at least one object that has a non-zero possibility of being in each cluster (the empty cluster cannot exist). An interesting property of these three types of partitions is that all crisp partitions are fuzzy partitions and all fuzzy partitions are possibilistic partitions. Equations (1.1), (1.2), and (1.3), show that $\mathbf{M}_{hcn} \subset \mathbf{M}_{fcn} \subset \mathbf{M}_{pcn}$.

All cluster analyses address the same questions, independent of the type of partition. The three main questions are: i) *cluster tendency*—how many clusters are there?; ii) *partitioning*—which objects belong to which cluster and to what degree?; and iii) *cluster validity*—are the partitions "good"? There are many algorithms that attempt to answer these questions. I focus on all three of these problems for the case in which the objects are described by (dis)similarity data.

1.2 Contributions

Clustering objects described by relational data represent a crucial and pertinent problem. Numerous real-world data sets, including ontologies, web documents, patient records, and many forms of bioinformatics data can only be represented as (dis)similarity data. Most of the discussion in this dissertation is tailored to square dissimilarity data; however, Sections 2.3 and 4.3 will specifically address clustering in rectangular relational data.

1.2.1 Analysis

Chapter 3 begins with a thorough description of a special subset of crisp partitions, called *aligned partitions*. A proof is presented that shows that the cardinality of the set of all crisp c-partitions $\mathbf{M}_{hcn}$ is much greater than the cardinality of the set of all aligned c-partitions $\mathbf{M}^*_{hcn}$. This leads to an analysis that shows that there is a direct relationship between the *Visual Assessment of cluster Tendency* (VAT) algorithm, *single-linkage* (SL) hierarchical clustering, and *Dunn's cluster validity index*. I also develop and prove a recursive formulation of the iVAT path-based distance transform. This formulation reduces the complexity of the iVAT algorithm from $O(n^3)$ to $O(n^2)$. These analyses provide a basis for the algorithms presented in Chapter 4.

1.2.2 Clustering in Ordered Dissimilarity Data

In Chapter 4, two novel algorithms are developed that attempt to answer the three important clustering questions for relational data. The *CLustering in Ordered Dissimilarity Data* (CLODD) algorithm computes aligned c-partitions in ordered dissimilarity data. A fitness function is developed on the basis of three heuristics of cluster validity: i) contrast between on-diagonal and off-diagonal blocks in the ordered dissimilarity matrix; ii) "edginess" at the boundaries of the on-diagonal blocks; and, iii) minimum allowable cluster size. *Particle Swarm Optimization* (PSO) is used to optimize the fitness function to find the "best" aligned c-partition of the VAT-ordered dissimilarity data. CLODD performs the operations of cluster tendency, partitioning, and cluster validity simultaneously to produce, for the case of VAT-ordered data, SL-type clusters.

1.2.3 New Formulation of co-VAT

The VAT algorithm adapted to rectangular relational data is called co-VAT [Bezdek et al., 2007]. The first step of co-VAT is to estimate a square dissimilarity matrix. I propose new statistics-based estimation method. I also present a new reordering formulation of co-VAT in Chapter 4. The iVAT path-based distance transform [Wang et al., 2010, Fisher et al., 2001] is shown to improve the contrast in VAT images. I adapt this transform to co-VAT which shows dramatic improvement in showing clustering tendency of "tough" cases. I call this algorithm co-iVAT.

1.2.4 Rectangular Single-Linkage

The second algorithm developed in Chapter 4 is *Rectangular Single-Linkage* (ReSL) clustering. CLODD (and other algorithms) is extended to partition the objects represented by rectangular data. ReSL clustering finds four types of partitions in these data: partitions of the row objects, partitions of the column objects, partitions in the union of the two sets of objects, as well as what we call co-clusters. To date, ReSL is the *only* algorithm that does this. And I show that ReSL is more effective than a leading co-clustering algorithm for a number of example data sets.

1.2.5 Ontological Self-Organizing Map

Chapter 5 presents a novel extension to Kohonen's *Self-Organizing Map* (SOM) that allows one to use SOMs with ontological data. The *Ontological Self-Organizing Map* (OSOM) uses a fuzzy prototype, where each dimension of the prototype vector represents a single term. Hence, each prototype element is the membership of the corresponding term in the prototype "sentence". Given a set of ontological training data, the OSOM produces a two-dimensional map of the high-dimensional data. This mapping is used to visualize the cluster tendency of the training data and to classify new inputs. The OSOM can also be used to produce summarizations of the mapped ontological data. The OSOM is demonstrated on sets of genes that are annotated by the *Gene Ontology* (GO). Comparisons with other SOM-based algorithms are also shown.

1.3 Dissertation Outline

This chapter has given an introduction to the problem. Additionally, my dissertation's contributions to clustering in relational data were introduced and a perspective on the difficulties was presented.

Chapter 2 presents previous work on relevant approaches to relational clustering. A broad review of relational clustering work is given and an in-depth look at similar work is presented. Specifically described are techniques that address cluster tendency, partitioning, and cluster validity for relational data. Also discussed are clustering in rectangular data and ontological data.

Chapters 3, 4, and 5 present the novel contributions of this dissertation. In Section 3.1, *aligned partitions* are presented. Sections 3.2 and 3.3 prove the relationship between VAT, SL, and Dunn's index. Also presented are illustrative numerical examples. In Section 3.4, a recursive formulation of the iVAT algorithm is presented. The analysis performed in Chapter 3 is important as it provides a basis for the algorithms developed in subsequent chapters.

Section 4.1 presents the framework of the CLODD algorithm. This clustering method combines visual clustering techniques, swarm optimization, and image processing to extract partitions from ordered dissimilarity data. Several experiments, with both simulated and real data, are shown that reveal the motivation behind CLODD as well as its strengths and limitations. The methodology behind CLODD is the underpinning of ReSL, which is presented in Section 4.3. The ReSL algorithm computes SL partitions of rectangular relational data. This claim is supported with analysis based on the results of the analysis in Chapter 3. Additionally, several numerical examples are shown that illustrate the results of ReSL and compare it to existing rectangular clustering algorithms. A preprocessing step of ReSL is co-VAT. Section 4.2 presents new formulations of coVAT, including co-iVAT.

Clustering in ontologies is a special case of relational clustering. Section 5.1 discusses the development of the ontological self-organizing map, OSOM. This algorithm is a novel extension of the SOM and is designed to provide visualization and summarization of ontological clusters. We apply the OSOM to genes annotated by GO terms and present results for various data sets.

Chapter 6 concludes this dissertation with a summary of open problems and ideas for future work.

1.4 List of Relevant Publications

The research described in this dissertation is based on material from the following publications:

1. T.C. Havens, J.M. Keller, and M. Popescu (2010). Computing with words with the ontological self organizing map. *IEEE Trans. Fuzzy Systems.*

2. T.C. Havens, J.C. Bezdek, and J.M. Keller (2010). A new implementation of the co-VAT algorithm for visual assessment of clusters in rectangular relational data. *Proc. ICAISC.*

3. T.C. Havens, J.C. Bezdek, J.M. Keller, M. Popescu, and J.M. Huband (2010). Is VAT really single linkage in disguise?. *Annals of Mathematics and Artificial Intelligence, 55*(3), 237-251.

4. T.C. Havens, J.C. Bezdek, J.M. Keller, and M. Popescu (2009). Clustering in ordered dissimilarity data, *International Journal of Intelligent Systems, 24*(5), 504-528.

5. T.C. Havens, J.C. Bezdek, J.M. Keller, and M. Popescu (2008). Dunn's cluster validity index as a contrast measure of VAT images. *Proc. ICPR*, 1-4.

6. T.C Havens, C.J. Spain, N.G. Salmon, and J.M. Keller (2008). Roach infestation optimization. *Proc. IEEE SIS*, 1-7.

7. T.C. Havens, J.M. Keller, M. Popescu, and J.C. Bezdek (2008). Ontological self-organizing feature maps for cluster visualization and functional summarization of gene products using Gene Ontology similarity measures. *Proc. FUZZ-IEEE*, 104-109.

8. T.C. Havens, J.M. Keller, E. MacNeal Rehrig, H.M. Appel, M. Popescu, J.C. Schultz, and J.C. Bezdek (2008). Fuzzy cluster analysis of bioinformatics data composed of microarray data and Gene Ontology annotations. *Proc. NAFIPS*, 1-6.

Chapter 2

State of the Art

Clustering is the process of grouping objects or patterns in a sensible manner. This process is often performed to elucidate the similarity and dissimilarity among and between the grouped objects. Clustering has also been called unsupervised learning, numerical taxonomy, typology, and partitioning [Theodoridis and Koutroumbas, 2009]. Although clustering is typically thought of as only the act of separating objects into the proper groups, cluster analysis actually consists of three concise questions: i) *cluster tendency*—how many clusters are there? ii) *partitioning*—which objects belong to which cluster and to what degree? iii) *cluster validity*—are the partitions "good"? Although most clustering work concentrates on the question of partitioning, I focus on all three of these issues.

Context plays an important part in clustering as the following example shows. Consider a set of objects $\mathbf{O} = \{o_1, \ldots, o_n\}$. In this example, the objects are the Ford Mustangs shown in Fig. 2.1. One can imagine that these cars can be grouped according to any number of criterion. For example, if one grouped the cars according to model year, the partition is $\{o_1, o_2\}, \{o_3\}, \{o_4, o_6\}, \{o_5\}$. The cars could also be grouped by the number of engine cylinders, yielding the partition $\{o_1, o_2, o_3, o_4, o_5\}, \{o_6\}$, where o_6, the Green 1978 Mustang II, is the only car pictured that does not have a V8 engine (it has a paltry, by comparison, V6). Figure 2.2 shows other example partitions of the cars, each based on a different *cluster criterion*. Moreover, all the cars shown in Fig. 2.1 could be grouped together according to Make-Model, albeit this grouping, which only produces one group, is trivial and reveals little about the set as a whole.

As Fig. 2.2 shows, the partitioning is directly dependent on the cluster criterion. The partitions shown in Fig. 2.2 are crisp partitions and are, hence, in $\mathbf{M}_{hcn}$, which is constructed in eq.(1.1). The crisp partition $\mathbf{U}$ that would represent the partition shown in Fig. 2.2(a) is

$$\mathbf{U} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix},$$

where $u_{ik} = 1$ indicates that the kth object is in the ith cluster. For this example, the first row of $\mathbf{U}$ is the cluster that represent the model-year 1966, the second row is the cluster that represent the model-year 1968, the third row represent the model-year 1973, and the last represent 1978. This example shows that the results of clustering depends on how one chooses to measure similarity between the objects. Hence, the context of the clustering problem is encapsulated in the distance or similarity measure chosen to measure the object-to-object relations.

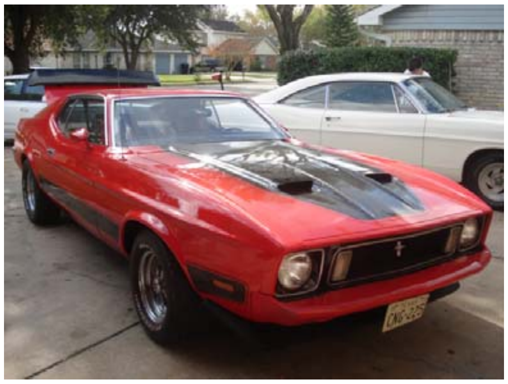
There are many algorithms that extract crisp partitions from unlabeled object sets; K-means [Hartigan, 1975, Hartigan and Wong, 1979] and hierarchical clustering [Johnson, 1967] being, arguably, the most popular. In general, there are three main types of partitioning algorithms [Theodoridis and Koutroumbas, 2009]: *sequential*—clusters are produced by sequentially iterating through the data; *hierarchical*—partitions are

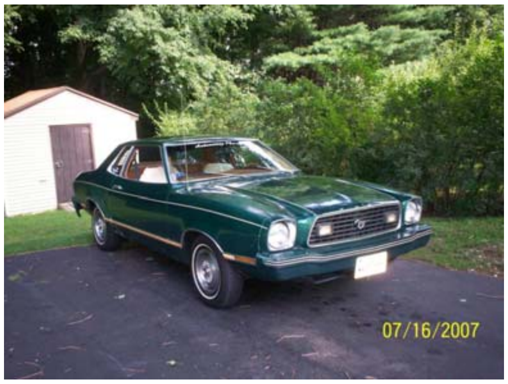
(a) o_1 - Black 1966 Coupe

(b) o_2 - Black 1966 Coupe

(c) o_3 - Red 1968 Convertible

(d) o_4 - White Red-Striped 1978 II Cobra


(e) o_5 - Red Black-Striped 1973 Mach 1



(f) o_6 - Green 1978 II

Figure 2.1: Which Ford Mustangs belong together?

1966	1968	1973	1978
o_1	o_3	o_5	o_4
o_2			o_6

(a) Model Year

6 Cylinders	8 Cylinders
o_6	o_1
	o_2
	o_3
	o_4
	o_5

(b) Number of Engine Cylinders

60's	70's
o_1	o_4
o_2	o_5
o_3	o_6

(c) Model Decade

Coupe	Fastback	Convertible
o_1	o_4	o_3
o_2	o_5	
o_6		

(d) Body Style

Black	Red	White	Green
o_1	o_3	o_4	o_6
o_2	o_5		

(e) Body Color

60's, 8 Cylinders	70's, 8 Cylinders	70's, 6 Cylinders
o_1	o_4	o_6
o_2	o_5	
o_3		

(f) Model Decade and Number of Cylinders

Figure 2.2: Resulting partitions of Ford Mustangs for different cluster criterions

agglomerated or divided to produce a hierarchy of clusters; and *partitional*—typically an objective function is optimized to produce a partition. Other types of partitioning algorithms include swarm-based algorithms [Handl et al., 2005, 2003, Chen et al., 2004], genetic clustering [Sheikh et al., 2008], and graph-theoretic methods [Hubert, 1974, Harary, 2004]. Partitioning algorithms are described in more detail in Section 2.2.

The color-based partition, in particular, in Fig. 2.2(e) outlines the strength of non-crisp partitions. The Mustangs in Figs. 2.1(d,e) are multi-colored; hence, the red with black-stripe Mustang in (e) could be considered to be part of the black partition and the white with red-stripe Mustang in (d) could be considered to be part of the red partition. However, it is clear that these two cars should not completely belong to either partition. Fuzzy and possibilistic partitions solves this conundrum by allowing objects to belong to more than one partition.

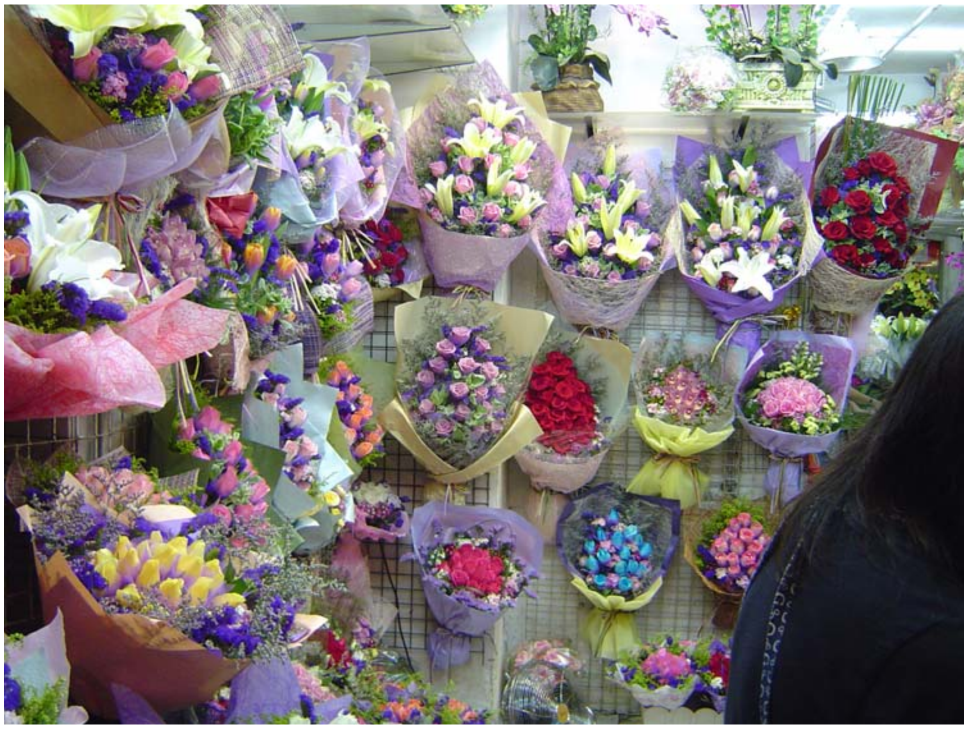


Figure 2.3: Flower stand in Hong Kong, China. By what criteria would you group these objects?

2.0.1 Fuzzy and Possibilistic Clustering

Figure 2.3 shows an image of a flower stand in Hong Kong, China. As the example shown in Figs. 2.1 and 2.2 illustrated, the number of clusters and partitions of the flowers in this image depend on the question asked. One could imagine that there are numerous crisp partitions of the flowers, based on the criteria such as type, number of petals, etc. However, there are criteria that result in ambiguous partitioning choices. For example, consider that the clustering criteria for these flowers is color. Color is a continuous spectrum and even flowers of the same type are not all exactly the same color. Thus, crisp partitions of the flowers on the basis of color could be misleading. Does a magenta flower belong in the red-cluster or the purple-cluster (or perhaps partially to the blue-cluster)? Fuzzy clustering allows the magenta flower to belong to more than one cluster to a certain degree.

Fuzzy and possibilistic clustering addresses the problem of ambiguity in clustering criteria by assigning a membership or typicality to each object. For fuzzy clustering, this membership indicates the degree to which an object belongs to each cluster. In possibilistic clustering, each object is assigned a typicality of belonging to a cluster.

2.0.2 Other Clustering Questions

There are other questions that are important to clustering including [Theodoridis and Koutroumbas, 2009]: *feature selection*—which features are important and how do features relate to each other?; *proximity measure*—what is the similarity or dissimilarity between features?; *partition interpretation*—what does the partition say about the data? Although these concerns are important to consider when applying clustering algorithms, I do not generally address them in this dissertation. I do, however, consider proximity measures for ontological data in Section 2.4.1. Good general resources on the topics of feature selection, proximity measures, and partition intrepretation include [Liu and Motoda, 1998, Liu and Yu, 2005, Guyon and Elisseeff, 2003, Theodoridis and Koutroumbas, 2009].

2.1 Visual Approaches to Clustering

For object data, visual clustering was initially performed by inspecting scatter-plots in $p = 1, 2$, and 3 dimensions. For $p > 3$, scatter-plots cannot be made. Many computational schemes have been devised to represent higher dimensional object data so that it can be visualized (and hence, possibly formed into clusters from visual representations). Interesting examples include Interactive Visual Clustering (ICV) [desJardins et al., 2007], HC-Enhanced [Tejada and Minghim, 2005], Andrews plots [Andrews, 1972], Chernoff faces [Chernoff, 1973], and Trees and Castles [Kleiner and

Hartigan, 1981]. There are many other approaches and the references [Tryon, 1939, Tukey, 1977, Everitt, 1978, Cleveland, 1993] contain informative introductions on many of these approaches.

For relational data $\mathbf{D}$, scatter-plots are unavailable. Tryon [1939] apparently presented the first method for extracting clusters from dissimilarity data by use of a visual approach. Here is a rough description of his method; (i) plot a graph of each row in the data—a matrix of pair-wise correlation coefficients, (ii) visually aggregate subsets of the graphs into clusters, (iii) reorder the input data matrix $\mathbf{D}$ so that similar profiles have adjacent representations in the rows and columns of the reordered data set $\mathbf{D}^*$, (iv) find the mean profile (a prototype graph representing the elements of a group) for each cluster of correlation profiles, and (v) present the final results as a set of clustered profile graphs with their prototypes. This procedure—almost 70 years old—contains all the elements of the current work on visual clustering: create a visual representation of $\mathbf{D}$, reorder it to $\mathbf{D}^*$, create a visual representation $\mathbf{D}^*$, and finally, extract clusters from $\mathbf{D}^*$ using the visual evidence. Tryon did this by hand in 1939 for a 20×20 data set collected at the University of California, Berkeley. For tiny data sets, methods such as these are useful. But for the data sets typically encountered today, automation is essential.

In the decades subsequent to Tryon's work, the literature has included many visual schemes for each of the three main problems in cluster analysis: tendency, partitioning, and validity. Using $\mathbf{D}$ and $\mathbf{D}^*$ in various ways for any of the three clustering problems involves two basic principles: *finding* $\mathbf{D}^*$ (how shall we reorder $\mathbf{D} \rightarrow \mathbf{D}^*$?), and *displaying* $\mathbf{D}^*$ (how shall we "see" the information in $\mathbf{D}^*$?). These three issues and two principles have appeared in almost every combination.

Sneath [1957] introduced the idea of visual representation of $\mathbf{D}^*$ by an image. Sneath's paper contains an image $I(\mathbf{D}^*)$ of $\mathbf{D}^*$ created by hand-shading the pixels of a matrix with one of eight "intensities"—reordering was done by an algorithm that had both computer and manual components. Subsequent refinements of his idea followed the general evolution of computers themselves. Floodgate and Hayes [1963] presented a hand rendered image similar to Sneath's, but reordering of $\mathbf{D}$ was done computationally using single-linkage clustering. Apparently Ling [1973] was the first to automate the creation of the image $I(\mathbf{D}^*)$ of $\mathbf{D}^*$ with an algorithm called SHADE, which was used after application of the complete linkage hierarchical clustering scheme and served as an alternative to visual displays of hierarchically nested clusters via the standard dendrogram. SHADE used 15 level halftone intensities (created by over-striking standard printed characters) to approximate a digital representation of the lower triangular part of the reordered dissimilarity matrix. SHADE apparently represents the first completely automated approach to finding $\mathbf{D}^*$ and viewing $I(\mathbf{D}^*)$.

Closely related to SHADE, but presented more in the spirit of finding rather than displaying clusters found with a relational clustering algorithm, is the "graphical method of shading" described in Johnson and Wichern [2007]. They provide this informal description: (i) arrange the pair-wise distances between points in the data into several classes of 15 or fewer, based on their magnitudes, (ii) replace all distances in each class by a common symbol with a certain shade of gray, (iii) reorganize the distance matrix so that items with common symbols appear in contiguous locations along the main diagonal (darker symbols correspond to smaller distances), and (iv) identify groups of similar items by the corresponding patches of dark shadings. A more formal approach to this problem is the work of Tran-Luu [1996], who proposed reordering the data into an "acceptable" block form based on optimizing several mathematical criteria of image "blockiness". The reordered matrix is then imaged and the number of clusters is deduced visually by a human observer.

Software for visualizing distance data is available at the GENLAB toolbox website [van Someren et al., 2000]. Similarity-based intensity images, formed using kernel functions, have been use in Girolami [2002] and Zhang and Chen [2003] to provide guidance in determining the number of clusters (tendency assessment, in spirit of the VAT algorithm), but no useful ordering scheme is offered there to facilitate the approach. Other representative studies include Baumgartner et al. [2001, 2000], Strehl and Ghosh [2000a,b], and Dhillon et al. [1998]. Visual cluster validity includes the work presented in Hathaway and Bezdek [2003] and Huband and Bezdek [2008].

An algorithm that is provided with Matlab is the *Reverse Cuthill-Mckee* (RCM) algorithm [George and Liu, 1981]. This algorithm attempts to minimize the bandwidth of a symmetric matrix to move non-zero elements closer to the diagonal. This is essentially the goal of cluster tendency visualization. However, RCM only works with binary matrices—e.g., graph connection matrices. Dissimilarity matrices can easily be converted to binary matrices by thresholding. However, RCM is very sensitive to this threshold value and, for this reason, it is not a good method for determining cluster tendency.

The main difference between the algorithms and methods described in this section and CLODD is that CLODD is a completely autonomous method for determining cluster tendency, extracting clusters from the image of the reordered dissimilarity data, and providing a cluster validity metric, as well. To my knowledge, there are no other methods, at the time of this dissertation, that attempt this trifecta. Furthermore, this leads to a distinct advantage of CLODD; namely, that CLODD is not tied directly to any one distance metric or reordering scheme. CLODD requires as input only an image of reordered dissimilarity data, such that the clusters appear as dark blocks along the diagonal.

2.1.1 Visual Assessment of cluster Tendency (VAT)

The VAT algorithm displays an image of reordered and scaled dissimilarity data [Bezdek and Hathaway, 2002]. Each pixel of the grayscale VAT image $I(\mathbf{D}^*)$ displays the scaled dissimilarity value of two objects. White pixels represent high dissimilarity, while black represents low dissimilarity. Each object is exactly similar with itself, which results in zero-valued (black) diagonal elements of $I(\mathbf{D}^*)$. The off-diagonal elements of $I(\mathbf{D}^*)$ are scaled to the range $[0, 1]$. A dark block along the diagonal of the $I(\mathbf{D}*)$ is a sub-matrix of "similarly small" dissimilarity values; hence, the dark block represents a cluster of objects that are relatively similar to each other. Thus, the cluster tendency is shown by the number of dark blocks along the diagonal of the VAT image.

The VAT algorithm is based on *Prim's algorithm* for finding the *Minimal Spanning Tree* (MST) of a weighted connected graph [Bezdek and Hathaway, 2002]. Algorithm 2.1.1 illustrates the steps of the VAT algorithm; notice the almost line-for-line similarity to Prim's algorithm, outlined in Algorithm 2.1.2. VAT reorders the dissimilarity matrix in the same order in which PA adds vertices to the MST. Already, it can be seen that there is a direct relation between VAT and the MST, which is further addressed in Section 3.2. The only differences between VAT and Prim's algorithm is the choice of the starting object (although Prim's algorithm could be initialized with the VAT starting object) and the end result; VAT produces an image, while Prim's algorithm produces a spanning tree.

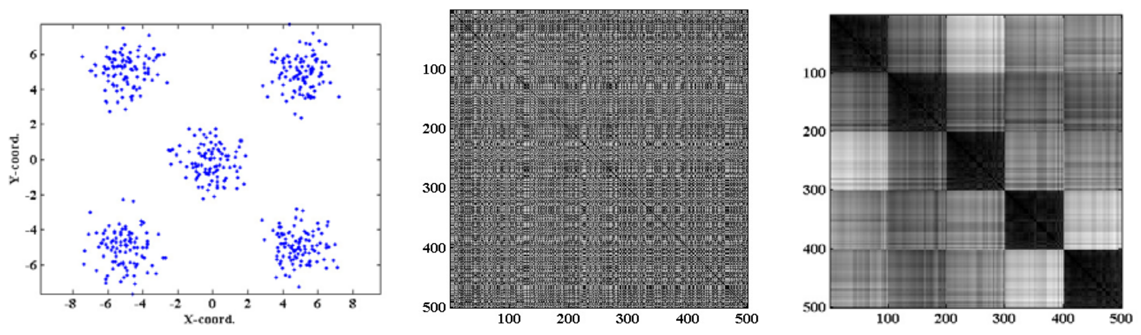
The resulting VAT-reordered dissimilarity matrix $\mathbf{D}^*$ can be normalized and mapped to a gray-scale image with black representing the minimum dissimilarity and white the maximum. Figure 2.4 is an example of the VAT image for a set of five clusters. The five dark blocks along the diagonal of the VAT image suggest that the object data seen in Fig. 2.4(a) possesses five clusters, but the clusters are not identified.

Algorithm 2.1.1: VAT Ordering Algorithm [Bezdek and Hathaway, 2002]

```
Input: $\mathbf{D}$ - dissimilarity matrix
Data: $\mathbf{K} = \{1, 2, \ldots, n\}$; $\mathbf{I} = \mathbf{J} = \emptyset$; $P = (0, 0, \ldots, 0)$.
Select
    $(i, j) \in \arg\max_{p \in \mathbf{K}, q \in \mathbf{K}} D_{pq}$.        (2.1)
Set $P(1) = i$; $\mathbf{I} = \{i\}$; and $\mathbf{J} = \mathbf{K} - \{i\}$.
for $r = 2, \ldots, n$ do
    Select
        $(i, j) \in \arg\min_{p \in \mathbf{I}, q \in \mathbf{J}} D_{pq}$.        (2.2)
    Set $P(r) = j$; Replace $\mathbf{I} \leftarrow \mathbf{I} \cup \{j\}$ and $\mathbf{J} \leftarrow \mathbf{J} - \{j\}$.
Obtain the ordered dissimilarity matrix $\mathbf{D}^*$ using the ordering array $P$ as:
$D^*_{pq} = D_{P(p),P(q)}$, for $1 \leq p, q \leq n$.
```

Algorithm 2.1.2: Prim's Algorithm [Prim, 1957]

```
Input: $\mathbf{D}$ - dissimilarity matrix
Data: $\mathbf{I} = \emptyset$; $\mathbf{J} = \{o_1, \ldots, o_n\}$
Pick a starting object $o_m$
$\mathbf{I} \leftarrow \{o_m\}$, $\mathbf{J} \leftarrow \mathbf{J} - \{o_m\}$
for $r = 2, \ldots, n$ do
    Select
        $(i, j) \in \arg\min_{o_p \in \mathbf{I}, o_q \in \mathbf{J}} D_{pq}$.        (2.3)
    Create an edge between $o_i$ and $o_j$.
    $\mathbf{I} \leftarrow \mathbf{I} \cup \{o_j\}$ and $\mathbf{J} \leftarrow \mathbf{J} - \{o_j\}$.
```



(a) Numerical object data - $\mathbf{X}$ (b) Unordered image - $I(\mathbf{D})$ (c) VAT image - $I(\mathbf{D}^*)$

Figure 2.4: Example of how VAT image suggests cluster tendency by the number of dark blocks along diagonal

There are many VAT-based algorithms that either extend the functionality of VAT or perform the functions of VAT automatically. These algorithms include, but are not limited to, *scalable VAT* (sVAT) [Hathaway et al., 2006], bigVAT [Huband et al., 2005], *revised VAT* (reVAT) [Huband et al., 2004], co-VAT [Bezdek et al., 2007], *Visual Cluster Validity* (VCV) [Hathaway and Bezdek, 2003, Huband and Bezdek, 2008, Ding and Harrison, 2007], *Correlation Cluster Validity* (CCV) [Popescu et al., 2008], *Dark Block Extraction* (DBE) [Wang et al., 2009], and *Cluster Count Extraction* (CCE) [Sledge et al., 2009a].

A VAT-based algorithm that is directly relevant to this dissertation is *improved*-VAT (iVAT) [Wang et al., 2010]. The iVAT algorithm uses a path-based distance measure from [Fisher et al., 2001]. Consider $\mathbf{D}$ to represent the weights of the edges of a fully-connected graph. The path-based distance is defined as

$$D'_{ij} = \min_{p \in P_{ij}} \max_{1 \leq h < |p|} D_{p[h]p[h+1]}, \tag{2.4}$$

where $p \in P_{ij}$ is an acyclic path in the set of all acyclic paths between vertex i (o_i) and vertex j (o_j), $p[h]$ is the index of the hth vertex along path p, and $|p|$ is the number of vertexes along the path. Hence, $D_{p[h]p[h+1]}$ is the weight of the hth edge along path p. Essentially the cost of each path p is the maximum weight of its $|p|$ edges. The distance between i and j is the cost of the minimum-cost path in P_{ij}. The authors of [Wang et al., 2010] first transform $\mathbf{D}$ into $\mathbf{D}'$ with (2.4), then they use VAT on the transformed dissimilarity matrix. The iVAT images show considerable improvement over VAT images in showing the cluster tendency for "tough" cases.

Computing $\mathbf{D}'$ directly from (2.4) can be thought of as a shortest path problem. The Floyd-Warshall algorithm [Cormen et al., 2009] is an algorithm that solves for all n^2 pairs of lowest-cost pathes in a connected graph with n nodes. The Floyd-

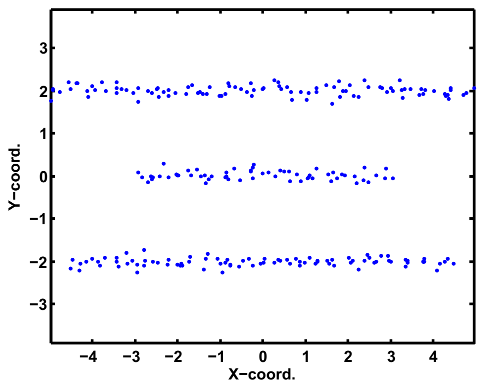
Warshall algorithm has a complexity of $O(n^3)$. The complexity of the VAT algorithm is $O(n^2)$; thus, the total complexity of iVAT as proposed in [Wang et al., 2010] is $O(n^3 + n^2) = O(n^3)$. I show in Section 3.4 that i) the iVAT dissimilarity matrix $\mathbf{D}'$ can be computed recursively from the VAT-reordered data $\mathbf{D}^*$ in $O(n^2)$.

Figure 2.5 illustrates the ability of iVAT to show correct cluster tendency for two cases where VAT "fails". Views (a,b) show object data from which a standard Euclidean norm is used to compute dissimilarity data $\mathbf{D}$. Views (c,e) show the respective VAT images of the *3 Lines* and *Boxes and Line* data. The VAT image of the *3 Lines* data clearly does not show the preferable tendency of 3 clusters. Although the VAT image of the *Boxes and Line* data shows the four tightly grouped clusters as 4 distinct dark blocks in the lower-right of the image, the large wavy cluster is not distinctly shown. In both cases, the iVAT images, shown in views (d,f), show the preferred cluster tendency—3 clusters in the *3 Lines* data and 5 clusters in the *Boxes and Line* data.

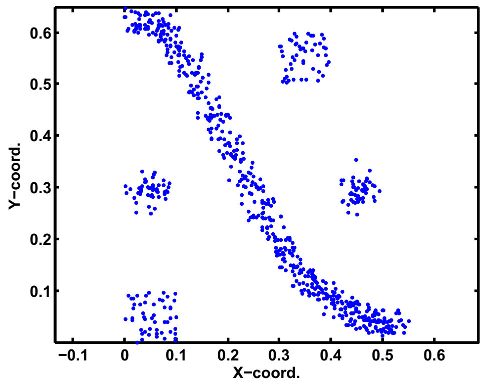
2.1.2 Other Block Reordering Methods

The reordering problem can be thought of as solving the *Traveling Salesman Problem* (TSP). Essentially, the ordering of the dissimilarity matrix is analogous to the permutation of the travel-plan that minimizes the distance a salesman would have to travel for a given set of connected map points. Map points that are close to one another should be visited together to minimize travel costs. There are a virtually uncountable number of proposed solutions to the TSP, from the most expensive, exhaustive methods, to more efficient methods.

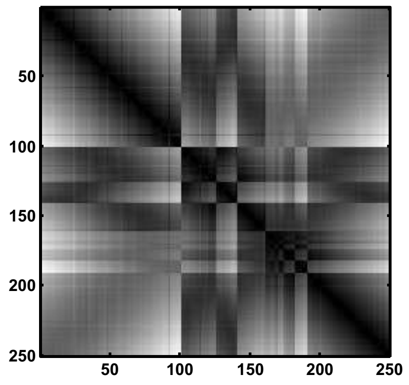
Rosenkrantz et al. [1977] and Reinelt [1994] compare several methods for solving the TSP. The descriptions of some of these schemes (translated into the context of matrix reordering) are outlined in Table 2.1 as well as the ratio of the trip cost to the



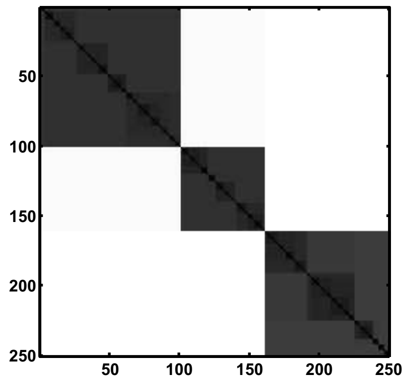
(a) 3 Lines Object Data



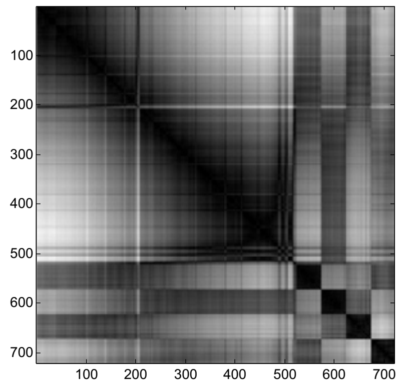
(b) Boxes and Line Object Data



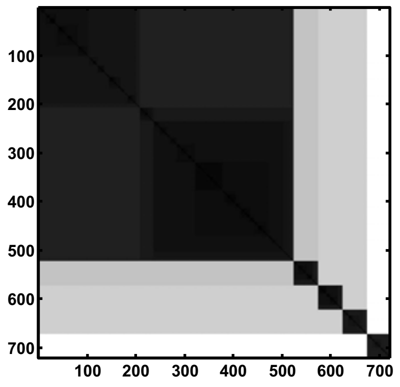
(c) 3 Lines VAT image - $I(\mathbf{D}^*)$



(d) 3 Lines iVAT image - $I(\mathbf{D}'^*)$



(e) Boxes and Line VAT image - $I(\mathbf{D}^*)$



(f) Boxes and Line iVAT image - $I(\mathbf{D}'^*)$

Figure 2.5: VAT and iVAT images of dissimilarity data where VAT “fails”

Table 2.1: Traveling salesman-based solutions to the matrix reordering problem. E/E^* indicates the ratio of the trip cost to the optimal trip cost. [Tran-Luu, 1996]

Method	Description
Furthest Insertion (FI)	The element that is most dissimilar to the current ordered set is inserted in the "best" location that minimizes E. $E/E^* \leq 1.25$
Cheapest Insertion (CI)	The element that minimizes E given the current ordered set is inserted in the "best" location that minimizes E. The complexity of this algorithm is $O(N^2 \log N)$. $E/E^* \leq 2(1 - 1/N)$
Nearest Insertion (NI)	The element most similar to any element in the current ordered set is inserted in the "best" location. The complexity of this algorithm is $O(N^2)$. $E/E^* \leq 2(1 - 1/N)$
Nearest Neighbor (NN)	The element most similar to the last added element in the ordered set is added. The complexity is $O(N^2)$. $E/E^* \leq 0.5\,(\text{ceil}(\log N) + 1)$

optimal trip cost, E/E^*. Notice that the initializations of the TSP-based algorithms are undefined. Hence, for comparison purposes, I will use the initial object chosen by VAT as the initial object for each of these reordering methods. How is the trip cost measured in the context of matrix reordering? Tran-Luu [1996] examined this problem.

The work of Tran-Luu [1996] focuses on reordering the dissimilarity matrix into an "acceptable block form" with the end goal of visualizing the clustering structure of these data. He proposes two measures of "blockness", the Bond Energy (BE) and the Linear Placement (LPm). However, as the author points out, the problems of optimizing these measures are NP-complete. Thus, the author continues by defining two heuristic solutions, linearization of the *Minimum Spanning Tree* (MST) and linearization of the dendogram generated by SL. The author's experimental results show that the LPm measure produces the best results when used with the linearization reordering methods as well as with the TSP-based solutions presented in Table 2.1. However, the LPm measure is computed for a dissimilarity matrix **D** by first thresholding $\mathbf{D}$ to obtain a binary matrix $\mathbf{B}$, $B_{ij} = \{1|D_{ij} \leq \sigma, 0|\text{else}\}$. Like Matlab's RCM method, the performance of this measure is very sensitive to the choice of the threshold. It should be noted that Tran-Luu [1996] did not specify or offer a method for setting the threshold of the LPm measure.

BE is computed as

$$E_{BE}(\mathbf{D}) = \sum_{i=1}^{N} \sum_{j=1}^{M} D_{ij} \cdot (D_{i-1,j} + D_{i+1,j} + D_{i,j-1} + D_{i,j+1}), \tag{2.5}$$

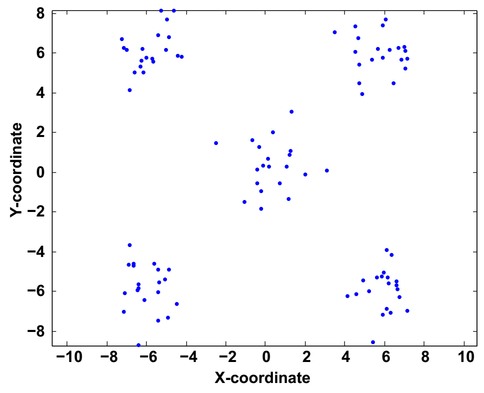
where $D_{0,j} = D_{N+1,j} = D_{i,0} = D_{i,M+1} = 0$. Equation 2.5 can be quickly computed in Matlab as

$$E_{BE}(\mathbf{D}) = \sum_{\forall i,j} D_{ij} \cdot (\mathbf{D} * \mathbf{A})_{ij},$$

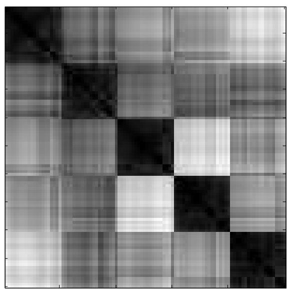
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

where $(*)$ indicates convolution.

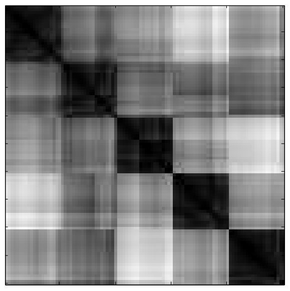
Figure 2.6 shows the *reordered dissimilarity images* (RDIs) computed using the methods described in this section on 100 objects that have a preferred (by me) cluster tendency of 5 clusters. Views (b-d) show that the VAT, NN, and FI algorithms show this preferred tendency, while the cluster tendency shown by the NI and CI algorithms in views (e,f) is unclear. Because the NI and CI algorithm are unable to effectively show the cluster tendency for this (seemingly) simply example, I will not show additional results for these algorithms.



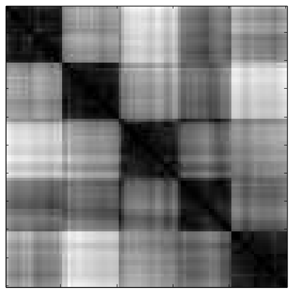
(a) Object Data



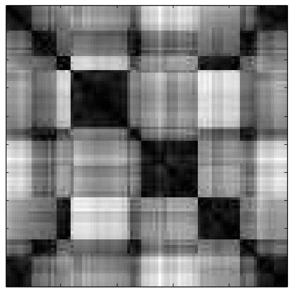
(b) VAT Image



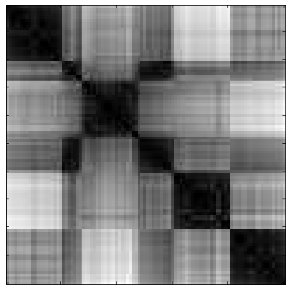
(c) NN Image



(d) FI Image



(e) NI Data



(f) CI Image

Figure 2.6: Reordered dissimilarity images of 5 Gaussian clouds

2.2 Partitioning Relational Data

There are many types of relational data. Figure 2.7 illustrates a few of these types. Figure 2.7(a) illustrates the type of object data that most clustering algorithms are designed for. For this datatype, the objects in $\mathbf{A}$ are described by a set of features, where $\mathbf{F}_A \in \mathbb{R}^{d \times n}$ denote the d-dimensional features of the n objects $\mathbf{O}$. In contrast to this type of problem, Fig. 2.7(b) illustrates the most well-known of relational data, square-relational, where only the relation between all pairs of objects are known. These data are denoted $\mathbf{R}_A \in \mathbb{R}^{n \times n}$, where $\mathbf{R}_A = [r_{ij} = \text{relation}(o_i, o_j) | 1 \leq i, j \leq n]$. This dissertation will primarily focus on relational data. These data are called homogeneous relations [Long et al., 2007]. As stated in Section 1.1, such relations typically represent the dissimilarity (or distance) of each pair of objects.

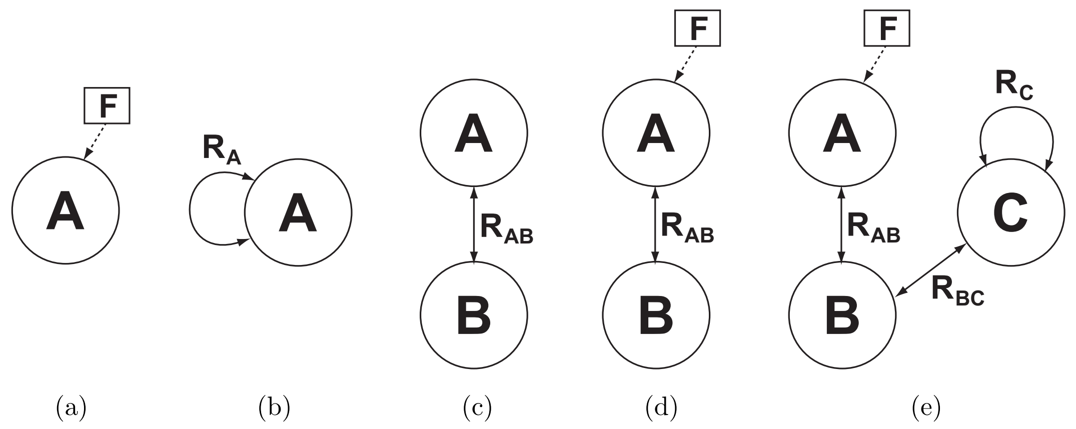


Figure 2.7: Different types of relational data. (a) Objects described by features (object data); (b) objects described by pair-wise relations (square relational data); (c) objects described by the relations to another set of objects (rectangular relational data); (d) and (e) show objects that are described by a combination of the types shown in (a), (b), and (c) [Long et al., 2007].

Figure 2.7(c) illustrates typical rectangular relational data. There are two sets of objects, $\mathbf{O}_A$ and $\mathbf{O}_B$ (often called the row and column objects), and only the relation is known between the row-column pairs, $\{o_{1,A}, o_{1,B}\}$, $\{o_{1,A}, o_{2,B}\}$, etc. The relational data among the row (or column) objects is not known. These data are denoted $\mathbf{R}_{AB} \in \mathbb{R}^{n_A \times n_B}$, where $\mathbf{R}_{AB} = [r_{ij} = \text{relation}(o_{i,A}, o_{j,B}) | 1 \leq i \leq n_A, 1 \leq j \leq n_B]$. If the objects in A and the objects in B are different types of objects (e.g. movies and reviewers, or genes and treatments), these data are called heterogeneous relations [Long et al., 2007]. Rectangular relational data are specifically addressed in Secs. 2.3 and 4.3.

Figures 2.7(d,e) show other types of relational data. View (d) illustrates that the data are composed of a set of feature vectors that describe $\mathbf{O}_A$ and a set of hetergeneous relations that describe the relationship between $\mathbf{O}_A$ and $\mathbf{O}_B$. View (e) illustrates a case where the date are composed of all types, including feature vectors, homogeneous relations, and heterogeneous relations. Although the data-types shown in Figs. 2.7(d,e) are interesting, I do not study these specifically herein. Another good reference on clustering mixed types of relational data is the work by Zhang et al. [2006].

2.2.1 Sequential Clustering

Sequential clustering algorithms are the most simple of all clustering algorithms. Essentially, the objects are presented sequentially and each object is either, i) partitioned into an existing cluster or ii) is partitioned into a new cluster. The *Basic Sequential Algorithmic Scheme* (BSAS) is perhaps the most basic of all clustering algorithms. BSAS is outlined in Algorithm 2.2.1. The cluster-to-object distance can be any chosen distance, e.g. SL, average-linkage, or complete-linkage (these distances are defined in Section 2.2.2).

Many variants exist to the BSAS, including modified BSAS (MBSAS) and the two-threshold scheme (TTSAS). The results of BSAS, in particular, are very dependent on the order in which the data are presented. MBSAS and TTSAS attempt to reduce this effect by deferring partition decisions for certain objects.

Sequential clustering algorithms are an attractive option for very-large data sets as these algorithms only iterate through the training data once; hence, they are very computationally inexpensive.

2.2.2 Hierarchical Clustering

Hierarchical clustering algorithms fall into two categories, agglomerative and divisive. Agglomerative algorithms create partitions by beginning with each object in its own cluster, or, in other words, with the singleton clusters, then partitions are built by combining clusters until all the objects are in one cluster. Divisive algorithms start with all objects in one cluster, the universal set, and then divide clusters until each

Algorithm 2.2.1: Basic Sequential Algorithmic Scheme [Theodoridis and Koutroumbas, 2009]

```
Data: Objects $\mathbf{O} = \{o_1, \ldots, o_n\}$
Choose distance threshold $\Theta$.
Start with $c = 1$ and $\mathbf{C}_1 = \{o_1\}$.
for $i = 2$ to $n$ do
    $j = \arg\min_k d(o_i, C_k)$
    if $d(o_i, C_j) \leq \Theta$ then
        $\mathbf{C}_j = \{\mathbf{C}_j, o_i\}$
    else
        $c \leftarrow c + 1$
        $\mathbf{C}_c = \{o_i\}$
```

object is in its own cluster (the singletons). Hierarchical clustering algorithms produce a partition for each value of $c = 1, \ldots, n$, where n is the total number of objects being partitioned.

Single-linkage

SL hierarchical clustering can be used on both numerical object data $\mathbf{X}$ and (square) relational data $\mathbf{D}$. I emphasize that $\mathbf{D}$ is square here to distinguish this problem from the case of *rectangular* relational data. Section 4.3 presents a rectangular SL algorithm (ReSL). In order to simplify the discussion of SL, I assume that numerical data has been converted to relational data by a vector norm distance measure. In terms of the dissimilarity matrix $\mathbf{D}$, the SL (set) distance between two clusters $\mathbf{C}_1$ and $\mathbf{C}_2$ is defined as

$$d_{SL}(\mathbf{C}_1, \mathbf{C}_2) = \min_{o_p \in \mathbf{C}_1, o_q \in \mathbf{C}_2} D_{pq}. \tag{2.6}$$

Figure 2.8 illustrates the SL distance between two clusters composed of two-dimensional numerical object data. This distance measure is used at each step of the SL agglomerative clustering algorithm to determine which two clusters are joined. Algorithm 2.2.2 outlines the steps of SL agglomerative clustering.

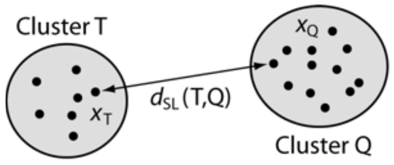


Figure 2.8: SL distance between two clusters

```
Algorithm 2.2.2: SL Agglomerative Clustering [Duda et al., 2000]
Input: D - dissimilarity matrix, $d_{pq} = \text{dissimilarity}(o_p, o_q),\ 1 \le i, j \le n$
Data: $\mathbf{C}^{(n)} = \{\{o_1\}, \{o_2\}, \ldots, \{o_n\}\}$
for $c = n, \ldots, 2$ do
    $\mathbf{S}_{ij} = d_{SL}(\mathbf{C}_i^{(c)}, \mathbf{C}_j^{(c)})\ \forall i, j$        (2.7)
    Find closest two clusters, $\{l, m\} = \arg\min_{i,j} \mathbf{S}_{ij}$.
    $\mathbf{C}^{(c-1)} \leftarrow \mathbf{C}^{(c)} - \mathbf{C}_l^{(c)} - \mathbf{C}_m^{(c)} + \mathbf{C}_l^{(c)} \cup \mathbf{C}_m^{(c)}$
```

SL clustering can also be expressed in terms of the MST of the connected graph that represent the object data. The weight of the edge between two objects is the dissimilarity value of those two objects. The MST is defined as the $n-1$ edges that fully connect the object data **O** and have the minimum summed edge weight [Harary, 2004]. A number of algorithms exist that compute the MST of a connected graph, Prim's and Kruskal's algorithms being two of the most popular [Prim, 1957, Kruskal, 1956]. The SL distance and the MST are related in that the weight of the MST edge between two subtrees of object data is the SL distance between them. Algorithm 2.1.2 outlines Prim's algorithm, where (2.3) returns the indices of the two closest objects from sets **I** and **J** according to the SL distance.

The SL clusters of the object data **O** can be found by cutting the high weight edges of the MST and examining the resulting subtrees [Gower and Ross, 1969]. For example, the 5-partition of **O** can be found by cutting the four highest weight edges in the MST of **O**, which results in five subtrees—each containing the objects that represent a cluster. I use this important result in Section 3.2.1 to prove that SL clusters are aligned partitions of the VAT reordered objects $\mathbf{O}^*$.

This dissertation will focus mostly on SL distance as it pertains to the research discussed. However, it is important to note that there are other distance measures that can be used with hierarchical clustering, such as average-linkage, complete-linkage, centroid-linkage, median-linkage, and Ward's-linkage. For the sake of completeness, average-linkage and complete-linkage are defined below.

Average- and complete-linkage

Average- and complete-linkage hierarchical clustering can be performed in the same manner as the SL agglomerative algorithm: Algorithm 2.2.2. However, Eq.(2.7) is replaced by the appropriate linkage distance. For a given c-partition, average-linkage is defined as

$$\begin{aligned} \mathbf{S}_{ij} &= d_{AL}(\mathbf{C}_i^{(c)}, \mathbf{C}_j^{(c)}) \\ &= \frac{1}{|\mathbf{C}_i||\mathbf{C}_j|} \sum_{o_p \in \mathbf{C}_i, o_q \in \mathbf{C}_j} D_{pq}, \end{aligned} \tag{2.8}$$

where $|\mathbf{C}_i|$ is the number of objects in the ith cluster. Complete-linkage is defined as

$$\begin{aligned} \mathbf{S}_{ij} &= d_{AL}(\mathbf{C}_i^{(c)}, \mathbf{C}_j^{(c)}) \\ &= \max_{o_p \in \mathbf{C}_i, o_q \in \mathbf{C}_j} D_{pq}. \end{aligned} \tag{2.9}$$

2.2.3 Partitional Clustering

In contrast to sequential and hierarchical clustering, partitional clustering directly computes the partitions by solving for an optima of a criterion function. The criteria typically involve optimizing the dissimilarity between clusters and the similarity among clusters. The most well-known crisp partitional clustering algorithms are k-

means and ISODATA [Ball and Hall, 1965]; the most well-known soft (fuzzy and possibilistic) clustering algorithms are *fuzzy c-means* (FCM) [Bezdek, 1981] and *possibilistic c-means* (PCM) [Krishnapuram and Keller, 1993]. There are numerous variants of these algorithms, some of which are described in Anderberg [1973], Hoppner et al. [1999], Jain and Dubes [1988], Pal et al. [1997], Timm et al. [2004], Timm and Kruse [2002], Pal et al. [2005], and Krisnapuram et al. [1995]. Many of these algorithms minimize an objective function of the form,

$$J(\mathbf{U}, \mathbf{V}) = \sum_{i=1}^{c} \sum_{k=1}^{n} U_{ik}^{m} \parallel \vec{x}_k - \vec{v}_i \parallel^2 - P(\mathbf{U}), \tag{2.10}$$

for a partition $\mathbf{U} \in \mathbf{M}_{hcn}$, $\mathbf{M}_{fcn}$, or $\mathbf{M}_{pcn}$, $\mathbf{V} = \{\vec{v}_1, \ldots, \vec{v}_c\} \in \mathbb{R}^{d \times c}$ are the cluster prototypes, c is the assumed number of clusters, and m is a "fuzzification" parameter. The function $P(\mathbf{U})$ is a penalty term that is often used in possibilistic clustering [Krishnapuram and Keller, 1993]. Equation (2.10) is often solved using an alternating optimization method, where, for the case of c-means algorithms, the cluster prototypes are computed by

$$\vec{v}_i = \sum_{k=1}^{n} U_{ik}^{m} \vec{x}_k / \sum_{k=1}^{n} U_{ik}^{m}, \tag{2.11}$$

where $m = 1$ for hard k-means and is the "fuzzification" parameter for fuzzy or possibilistic methods. For the k-means algorithm, the partition matrix is then computed by

$$U_{ik} = \begin{cases} 0 & D_{ik} > \min_{1 \le j \le c}\{D_{ik}\} \\ 1 & \text{else} \end{cases} . \tag{2.12}$$

Note that if there is a tie in the above equation, the object is partitioned into only one cluster. For the FCM algorithm, the partition update is

$$U_{ik} = 1 / \left[\sum_{j=1}^{c} (D_{ik}/D_{jk})^{2/(m-1)} \right]. \tag{2.13}$$

The prototype and partition updates are alternated until convergence (there is little change in $\mathbf{U}$). In relational data the quantity $\parallel \vec{x}_k - \vec{v}_i \parallel^2$ cannot be directly computed as the object data $\mathbf{X}$ and prototype vectors $\vec{v}$ are unavailable. Thus, the c-means algorithms cannot be directly applied to relational data.

Hathaway et al. [1989] proposed *relational* duals of the c-means algorithms, FCM and k-means by reformulating (2.10) as

$$K(\mathbf{U}) = \sum_{i=1}^{c} \left(\sum_{j=1}^{n} \sum_{k=1}^{n} (U_{ij}^m U_{ik}^m \delta_{jk}^2) / \left(2 \sum_{t=1}^{n} U_{it}^m \right) \right), \tag{2.14}$$

where δ_{jk}^2 is pair-wise squared-distance. The relational k-means algorithm and the relational FCM algorithm are outlined in Algorithms 2.2.3 and 2.2.4, respectively.

Algorithm 2.2.3: Relational hard k-means [Hathaway et al., 1989]

```
Input: $\mathbf{D}$ - squared-distance matrix, $d_{pq} = \text{dissimilarity}(o_p, o_q),\ 1 \le i, j \le n$, $c$
       - number of clusters
Data: $\mathbf{U}$ - partition matrix, where $U_{ij}$ is the membership of object $j$ in cluster
      $i$.
while $\parallel \mathbf{U}^{new} - \mathbf{U}^{old} \parallel \le \epsilon$ do
    Calculate the $c$ "prototypes" using $\mathbf{U}^{old}$,
        $$v_i = (U_{i1}, U_{i2}, \ldots, U_{in})^T / \sum_{k=1}^{n} U_{ik}. \quad (2.15)$$
    Update $\mathbf{U}$ using (2.12) with
        $$d_{ik}^2 = (\mathbf{D}\vec{v}_i)_k - (\vec{v}_i^T \mathbf{D} \vec{v}_i)/2. \quad (2.16)$$
```

Algorithm 2.2.4: Relational fuzzy c-means [Hathaway et al., 1989]

```
Input: D - squared-distance matrix, $d_{pq}$ = dissimilarity$(o_p, o_q)$, $1 \leq i, j \leq n$, $c$
       - number of clusters, $m$ - fuzzifier
Data: U - partition matrix, where $U_{ij}$ is the membership of object $j$ in cluster
      $i$.
while $\| \mathbf{U}^{new} - \mathbf{U}^{old} \| \leq \epsilon$ do
    Calculate the $c$ prototypes using $\mathbf{U}^{old}$,
        $v_i = (U_{i1}^m, U_{i2}^m, \ldots, U_{in}^m)^T / \sum_{k=1}^{n} U_{ik}^m.$    (2.17)
    Update U using (2.13) with
        $d_{ik}^2 = (\mathbf{D}\vec{v}_i)_k - (\vec{v}_i^T \mathbf{D}\vec{v}_i)/2.$
```

The relational c-means algorithms are limited to relational data that are Euclidean distance matrices, where these matrices are defined as as a symmetric, zero-diagonal matrix that can be expressed as

$$D_{ij}^2 = \| \vec{x}_i - \vec{x}_j \|_2^2, \ \vec{x} \in \mathbb{R}^D. \tag{2.18}$$

If $\mathbf{D}$ is not a distance matrix then the calculation of (2.16) can result in a negative value. In relational hard k-means, this is not necessary a bad thing as one merely compares distance values in (2.12). However, a negative distance in the calculation of the FCM partition update, (2.13), is disastrous. Hence, Hathaway and Bezdek [1994a] proposed *Non-Euclidean Relational Fuzzy c-Means* (NERFcM) as a way to generalize RFCM to pre-distance matrices ($n \times n$ symmetric matrices with non-negative elements and a zero-valued diagonal).

To overcome the problem of non-Euclidean data, NERFcM uses the β-spread transform to convert the non-Euclidean matrix $\mathbf{D}$ into a Euclidean matrix $\mathbf{D}_\beta$. The

β-spread transform is defined as

$$\mathbf{D}_\beta = \mathbf{D} + \beta\left(\mathbf{M} - \mathbf{I}\right), \tag{2.19}$$

where β is a scalar value, $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix, and $\mathbf{M} \in \mathbb{R}^{n\times n}$ is the diagonal matrix $M_{ij} = 1, i = j$ and $M_{ij} = 0, i \neq j$. The idea is to choose a "big-enough" value for β such that the values d_{ik}^2 computed by (2.16) are all non-negative. The NERFcM algorithm is essentially the same of the relational FCM algorithm, except that $\mathbf{D}_\beta$ is used in all equations. If following the distance calculation a negative d_{ik}^2 occurs, the matrix $\mathbf{D}_\beta$ is recalculated with a value of $\beta + \Delta\beta$, where

$$\Delta\beta = \max_{i,k} = \{-2d_{ik}^2 / \parallel \vec{v}_i - \vec{e}_k \parallel^2\}, \tag{2.20}$$

and $\vec{e}_k$ is the kth column of the identity matrix. Hathaway and Bezdek [1994b] and Hathaway and Bezdek [1994a] provide a proof that the β-spread transform accomplishes the task of transforming a non-Euclidean distance matrix to a Euclidean distance matrix. Note, however, that () is an approximation to this task.

2.3 Clustering in Rectangular Data

Rectangular object data (also called bi-type data) consist of m row-objects $\mathbf{O}_r$ and n column-objects $\mathbf{O}_c$, which are disjoint sets of objects, and, thus, comprise a set of $N = m + n$ total objects. Unlike square relational data, rectangular relational data only contain relation values between the disjoint object sets $\mathbf{O}_r$ and $\mathbf{O}_c$, leaving the inter-relations of objects in either $\mathbf{O}_r$ or $\mathbf{O}_c$ undefined. An example of rectangular relational data is the reviewer-ratings of movies, where the row-objects are movies, the column-objects are reviewers, and the ratings relate each reviewer to each movie.

The relation between reviewers (or movies) is unknown. Another example is document data, where the row-objects are documents, the column-objects are words or terms, and the relational data are the frequencies of each word in each document. Rectangular data are illustrated in Fig. 2.7(c).

VanMechelen et al. [2004] describes rectangular relational data as "proximity type", where in the rectangular case the data are considered incomplete because the full set of $N = m + n$ proximities do not exist. The reasons for the incompleteness could be due to the heterogenous nature of the data, such that (dis)similarity measures do not exist to measure the proximity between row objects and objects.

There are five types of clusters in rectangular data; Table 2.2 outlines these types [Bezdek et al., 2007]. The three types of clusters that are composed from the the union of the row-object and column-objects, $\mathbf{O}_r \cup \mathbf{O}_c$, are of special interest. Clusters of type P3 are composed of either $\mathbf{O}_r$ or $\mathbf{O}_c$, and these clusters are not mixed. Often the number of type P3 clusters in $\mathbf{O}_r \cup \mathbf{O}_c$ is $c_{r\cup c} = c_r + c_c$. Although P3 clusters are only composed of either row or column objects, there could be P3 clusters that are composed of row objects that are very similar to P3 clusters composed of column objects. If these clusters are joined, where row and column objects are mixed, these form the P4 clusters. These clusters, often called *co-clusters*, are the type that most bi-clustering or co-clustering algorithms look for. The P5 clusters are similar to P3 clusters in that they are only composed of either $\mathbf{O}_r$ or $\mathbf{O}_c$. However, these clusters do not include the objects that are in co-clusters. Hence, $c_p = (c_{r\cup c} - c_{co})$.

Example 2.3.1 illustrates the five cluster types for 2D Euclidean object data.

2.3.1 co-VAT: Assessing Tendency in Rectangular Data

The VAT algorithm for rectangular dissimilarity data is called co-VAT. The "co" prefix comes from the property of co-VAT in showing the number of co-clusters in

Table 2.2: Types of Clusters in Rectangular Dissimilarity Data [Bezdek et al., 2007]

Type	No. of Clusters	Description
P1	c_r	Clusters in row-objects $\mathbf{O}_r$
P2	c_c	Clusters in column-objects $\mathbf{O}_c$
P3	$c_{r\cup c}$	*All* clusters in the union of objects $\mathbf{O}_r \cup \mathbf{O}_c$
P4	c_{co}	*Mixed* clusters in the union of objects $\mathbf{O}_r \cup \mathbf{O}_c$
P5	c_p	*Pure* clusters in the union of objects $\mathbf{O}_r \cup \mathbf{O}_c$

rectangular data. The rectangular matrix $\mathbf{D}$ contains pair-wise dissimilarity values between m row-objects $\mathbf{O}_r$ and n column-objects $\mathbf{O}_c$. Assume the two sets of objects are indexed as $\mathbf{O}_r = \{o_1, \ldots, o_m\}$ and $\mathbf{O}_c = \{o_{m+1}, \ldots, o_{m+n}\}$. Thus the ith row and jth column entry of $\mathbf{D}$, $d_{ij} = d(o_i, o_{m+j})$.

co-VAT begins by creating a square matrix $\mathbf{D}_{r\cup c}$, part of which is composed of the rectangular dissimilarity matrix $\mathbf{D}$. $\mathbf{D}_{r\cup c}$ is created by, first, estimating the dissimilarity matrices $\mathbf{D}_r$ and $\mathbf{D}_c$, which are, respectively, square dissimilarity matrices that relate the objects in $\mathbf{O}_r$ and $\mathbf{O}_c$ to themselves — i.e. $[D_r]_{ij} \approx d(o_i, o_j)$ and $[D_c]_{ij} \approx d(o_{m+i}, o_{m+j})$. $\mathbf{D}_{r\cup c}$ is organized as

$$\mathbf{D}_{r\cup c} = \begin{bmatrix} \mathbf{D}_r & \mathbf{D} \\ \mathbf{D}^T & \mathbf{D}_c \end{bmatrix} = \tag{2.21}$$

$$= \begin{bmatrix} \begin{bmatrix} d(o_1,o_1) & \cdots & d(o_1,o_m) \\ \vdots & \ddots & \vdots \\ d(o_m,o_1) & \cdots & d(o_m,o_m) \end{bmatrix} & \begin{bmatrix} d(o_1,o_{m+1}) & \cdots & d(o_1,o_{m+n}) \\ \vdots & \ddots & \vdots \\ d(o_m,o_{m+1}) & \cdots & d(o_m,o_{m+n}) \end{bmatrix} \\ \begin{bmatrix} d(o_1,o_{m+1}) & \cdots & d(o_m,o_{m+n}) \\ \vdots & \ddots & \vdots \\ d(o_1,o_{m+n}) & \cdots & d(o_m,o_{m+n}) \end{bmatrix} & \begin{bmatrix} d(o_{m+1},o_{m+1}) & \cdots & d(o_{m+1},o_{m+n}) \\ \vdots & \ddots & \vdots \\ d(o_{m+n},o_{m+1}) & \cdots & d(o_{m+n},o_{m+n}) \end{bmatrix} \end{bmatrix}.$$

The elements in $\mathbf{D}_r$ and $\mathbf{D}_c$ are estimated from $\mathbf{D}$ using any vector norm on $\mathbb{R}^n$ and

$\mathbb{R}^m$,

$$[\mathbf{D}_r]_{ij} = \lambda_r||\vec{d}_{i*} - \vec{d}_{j*}||,\ 1 \leq i,j \leq m, \tag{2.22}$$

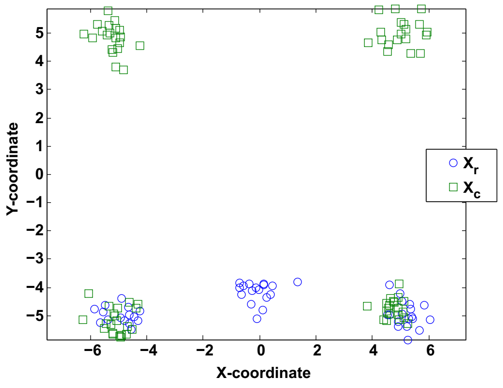
$$[\mathbf{D}_c]_{ij} = \lambda_c||\vec{d}_{*i} - \vec{d}_{*j}||,\ 1 \leq i,j \leq n, \tag{2.23}$$

where $\vec{d}_{i*}$ is the ith row of $\mathbf{D}$, $\vec{d}_{*j}$ is the jth column of $\mathbf{D}$, and λ_r and λ_c are scale factors such that the mean of the off-diagonal elements of $\mathbf{D}_r$ and $\mathbf{D}_c$ match the mean of $\mathbf{D}$. The scale factors λ_r and λ_c ensure that the relative scale of the dissimilarity values in $\mathbf{D}_r$ and $\mathbf{D}_c$ is the same as in $\mathbf{D}$, which is important for the construction of $D_{r\cup c}$ as this matrix is built from all three matrices, $\mathbf{D}_r$, $\mathbf{D}_c$, and $\mathbf{D}$. Because co-VAT is a visualization algorithm, the scale factors ensure that the imputed dissimilarity matrices $\mathbf{D}_r$ and $\mathbf{D}_c$ can be viewed with the same gray scale as $\mathbf{D}$.

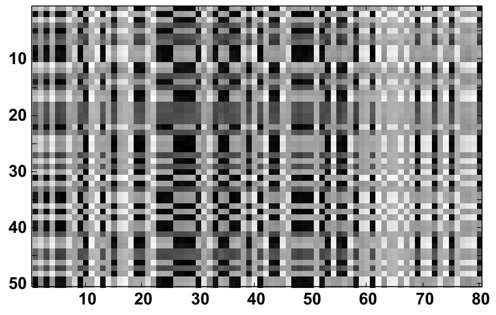
Algorithm 2.3.1 outlines the steps of co-VAT, which obtains a reordered rectangular matrix $\mathbf{D}^*$. By viewing the image $I(\mathbf{D}^*)$, one can determine the cluster tendency of the rectangular data. However, this algorithm is fundamentally different than VAT in that it displays the cluster tendency of the multiple types of rectangular data clusters. Note that line 6 in algorithm 2.3.1 is a corrected version of the original co-VAT definition in [Bezdek et al., 2007].

Example 2.3.1 (Three row-object clouds, Four column-object clouds). This example shows the five types of rectangular clusters types. Figure 2.9(a) shows a plot of 50 row-objects (circles) and 80 column-objects (squares). Note that objects in this example are Euclidean (for demonstration purposes); however, this is usually not the case for objects represented by rectangular relational data.

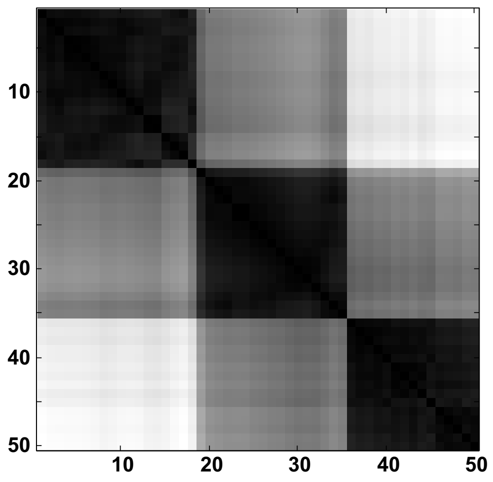
The P1 clusters are the three clusters of row-objects organized at the bottom of the plot. The P2 clusters are the four clusters of column-objects at the four corners of the plot. There are seven clusters in the union of the objects, namely the three P1



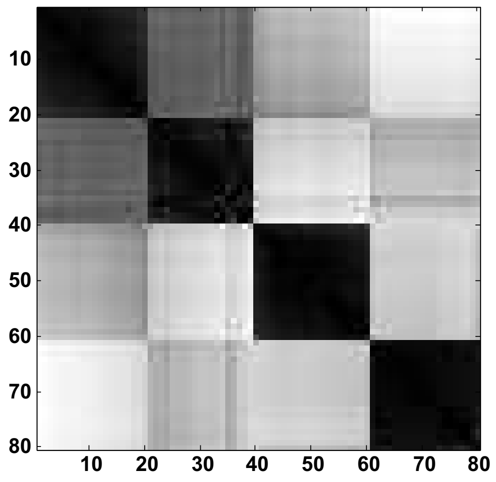
(a) $\mathbf{X}_r$ - 50 row objects, $\mathbf{X}_c$ - 80 column objects



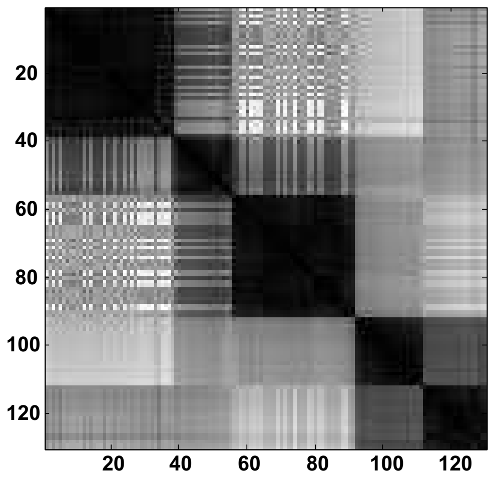
(b) Dissimilarity data



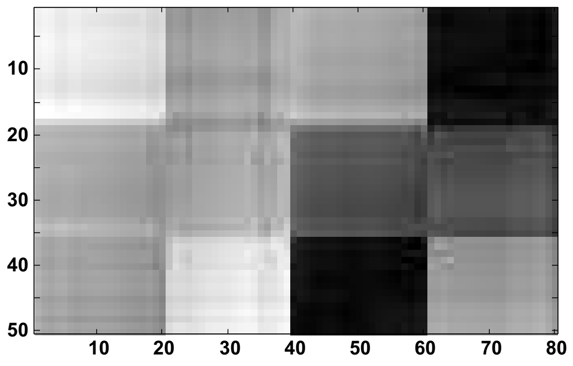
(c) $\mathbf{D}_r^*$



(d) $\mathbf{D}_c^*$



(e) $\mathbf{D}_{r \cup c}^*$



(f) $\mathbf{D}^*$

Figure 2.9: Rectangular relational data example. (a,b) - object data and dissimilarity data; (c-f) - co-VAT-reordered dissimilarity data matrices.

Algorithm 2.3.1: co-VAT Algorithm [Bezdek et al., 2007]

```
Input: $\mathbf{D}$ - $m \times n$ rectangular dissimilarity matrix
Build estimates of $\mathbf{D}_r$ and $\mathbf{D}_c$ using Eqs.(2.22) and (2.23), respectively.
Build $\mathbf{D}_{r \cup c}$ using Eq.(2.21).
Run VAT on $\mathbf{D}_{r \cup c}$, saving permutation array, $P_{r \cup c} = \{P(1), \ldots, P(m+n)\}$
Initialize $rc = cc = 0$; $RP = CP = 0$.
1 for $t = 1, \ldots, m+n$ do
2     if $P(t) \leq m$ then
3         $rc = rc + 1$, $rc$ is row component
4         $RP(rc) = P(t)$, $RP$ are row indexes
      else
5         $cc = cc + 1$, $cc$ is column component
6         $CP(cc) = P(t) - m$, $CP$ are column indexes
Form the co-VAT ordered rectangular dissimilarity matrix,
$\mathbf{D}^* = [d^*_{ij}] = [d_{RP(i)CP(j)}]$, $1 \leq i \leq m$; $1 \leq j \leq n$
```

clusters and the four P2 clusters. The P4 clusters (or co-clusters) are the five groups that are apparent if the row-objects and column-objects are mixed together. Finally, there are two P5 clusters, the two groups that are composed of both row and column objects at the bottom-left and bottom-right of the plot.

Figure 2.9(b) shows the rectangular dissimilarity data for the 50 row-objects and 80 column-objects in view (a). The tendency of three (P1) clusters in the row-objects and the four (P2) clusters in the column-objects is clearly shown in the VAT images of $\mathbf{D}^*_r$ and $\mathbf{D}^*_c$ in Figs. 2.9(c,d), respectively. The tendency of five (P4) clusters is shown in the VAT image of $\mathbf{D}^*_{r \cup c}$ in Fig. 2.9(e). Finally, the co-VAT image of the rectangular dissimilarity data is creating using the reordering indexes of $\mathbf{D}^*_{r \cup c}$. The co-VAT image is shown in view (f) and shows a tendency of two clusters that are composed of both row and column objects.

Example 2.3.2 (Pure rectangular relational data). This example shows the utility of co-VAT on relational data that are not derived from object data (as was shown in the previous example). Figure 2.10(a) shows a dissimilarity matrix of 250 row objects and

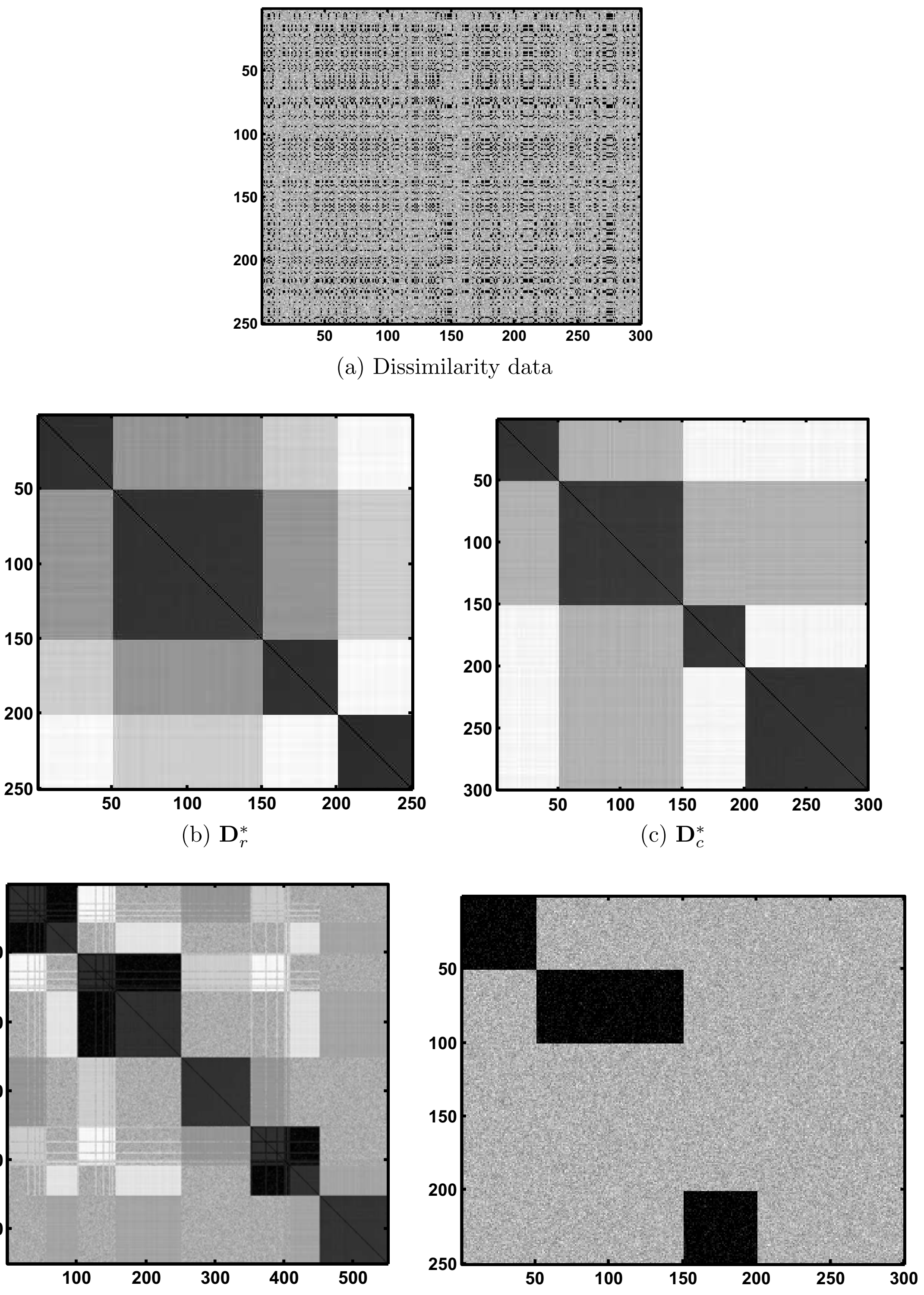
300 column objects. Figures 2.10(b-e) show the co-VAT images of this dissimilarity data. View (b) indicates that there are 4 (P1) clusters of row objects; view (c) indicates that there are 4 (P2) clusters of column objects. View (d) illustrates that in the union of the row and column objects there are 5 (P4) co-clusters. Finally, view (e) is the co-VAT image, which shows a tendency of 3 (P5) clusters.

Next, I move on to clustering in another type of data: *ontologies*.

2.4 Clustering in Ontological Data

Ontological data is composed of collections of terms organized in a hierarchical taxonomy. I refer to these data sets as ontological data. These are typically composed of hundreds of dimensions (individual terms) and can also be very large—on the order of 10,000 samples. One example of ontological data is the *Gene Ontology* (GO) [The Gene Ontology Consortium, 2004]. The GO is a hierarchical taxonomy of characteristic annotations of genes. Each term in the GO is taken from a controlled vocabulary, or corpus, and describes gene and gene product attributes. Although the ontology-based examples presented in this dissertation are based on the GO, it is easy to generalize any of the algorithms presented here to any ontology, of which numerical pair-wise term similarities can be generated. Examples of such ontological data include patient medical records, stock nomenclature, and web documents.

Suppose two genes, G_1 and G_2, are represented by a set of GO terms $G_1 = \{T_{11}, T_{12}, \ldots, T_{1n}\}$ and $G_2 = \{T_{21}, T_{22}, \ldots, T_{2m}\}$. For these sets, one can compute a similarity value using a number of methods [see Lord et al., 2003, Jiang and Conrath, 1997, Keller et al., 2004, Popescu et al., 2006, Keller et al., 2006]. These methods are described in detail in Section 2.4.1. Examples of GO annotations for genes that are used in this dissertation are presented in Tables 2.3 and 2.5. Table 2.3 shows the GO


(a) Dissimilarity data

(b) $\mathbf{D}_r^*$

(c) $\mathbf{D}_c^*$

(d) $\mathbf{D}_{r\cup c}^*$

(e) $\mathbf{D}^*$

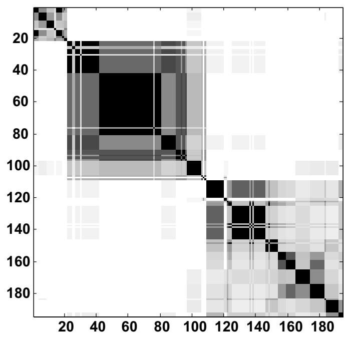
Figure 2.10: Pure rectangular relational data example. (a) - dissimilarity data; (b-e) - co-VAT-reordered dissimilarity data matrices.

Table 2.3: Example GO Annotations for $GPD194_{12.10.03}$ Proteins.

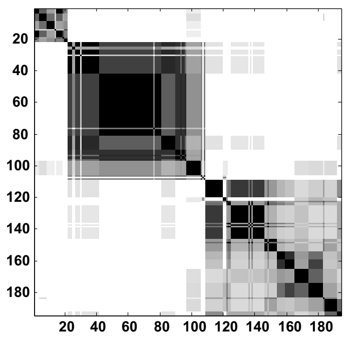
GO id	GO term	Definition
AAC79117	GO:0004722	protein serine/threonine phosphatase activity
(MTMR1)	GO:0004725	protein tyrosine phosphatase activity
	GO:0016787	hydrolase activity
	GO:0006470	protein amino acid dephosphorylation
	GO:0008372	cellular component
AAK34949	GO:0004713	protein tyrosine kinase activity
(FGFR4)	GO:0005524	ATP binding
	GO:0016740	transferase activity
	GO:0006468	protein amino acid phosphorylation
	GO:0004872	receptor activity
AAA51844	GO:0005201	extracellular matrix structural constituent
(COL1A2)	GO:0005581	collagen
	GO:0008147	structural constituent of bone
	GO:0001501	skeletal system development
	GO:0005584	collagen type I

annotations of 3 human gene products selected from a well-studied data set, called $GPD194_{12.10.03}$. The $GPD194_{12.10.03}$ data set contains 194 human gene products that appear in the GO. Popescu et al. [2004] contains a detailed description of the construction of this data. In brief, the object data are comprised of 21 gene products from the *myotublarin* protein family, 87 gene products from the *receptor precursor* protein family, and 86 gene products from the *collagen alpha chain* protein family. Table 2.4 presents the ENSEMBL characteristics of the $GPD194_{12.10.03}$ families. The three protein families are clearly visible in Fig. 2.11, which shows images of $\mathbf{D}_{194}$ (built with various GO-based similarity measures) ordered according to family. Note the strong substructure within the *collagen alpha chain* protein family, which is the block at the bottom right of the images in Fig. 2.11. This substructure has been corroborated by Myllyharju and Kivirikko [2004] and is also supported by the algorithms in this dissertation.

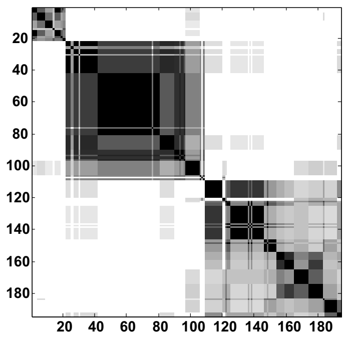
Figure 2.12 shows VAT-reordered dissimilarity data $\mathbf{D}^*$ for a set of 198 Arabidopsis



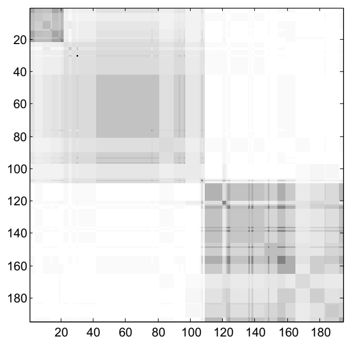
(a) Jaccard



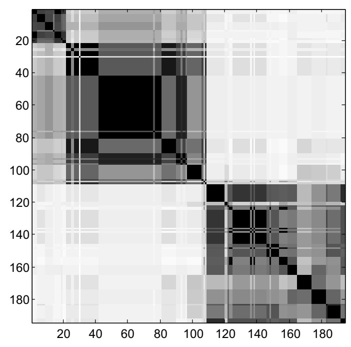
(b) Dice



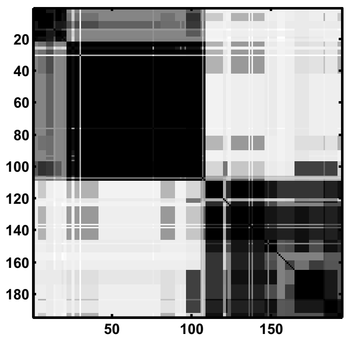
(c) Cosine



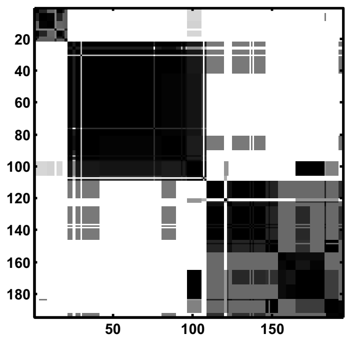
(d) Average IC



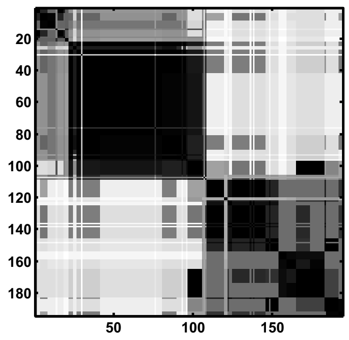
(e) Normalized Average IC



(f) OWA(4) IC



(g) Fuzzy Measure



(h) Augmented Fuzzy Measure

Figure 2.11: Dissimilarity data $\mathbf{D}_{194}$ of $GPD194_{12.10.03}$ built with different similarity measures, $\mathbf{D} = 1 - \mathbf{S}$.

Table 2.4: Characteristics of the $GPD194_{12.10.03}$ Data Set Extracted rom ENSEMBL [Hubbard et al., 2009].

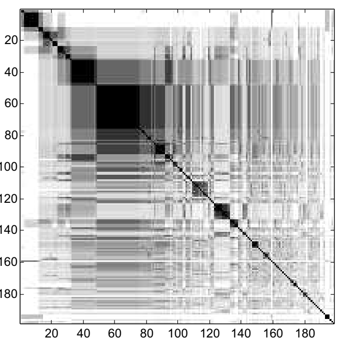
ENSEMBL family ID	F_i = Protein family	N_i = No. of sequences	Indices in Fig. 2.11
339	myotubularin	21	1-21
73	receptor precursor	87	22-108
42	collagen alpha chain	86	109-194

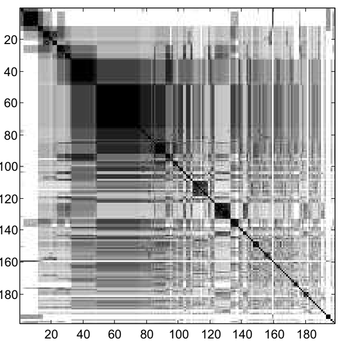
transcription factors — proteins that bind to regulatory regions and help control gene expression. The Gene Ontology annotations for four of these genes are shown in Table 2.5. In [Havens et al., 2008], we investigated methods for fusing the dissimilarity data shown in Fig. 2.12 with microarray data from an experiment that examined the genetic effects of insect and wounding stress.

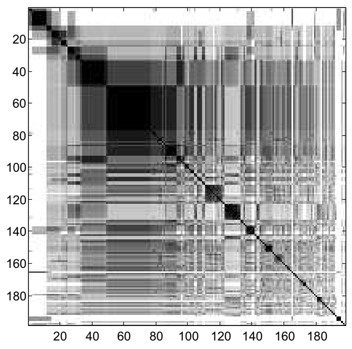
2.4.1 Ontology Similarity Measures

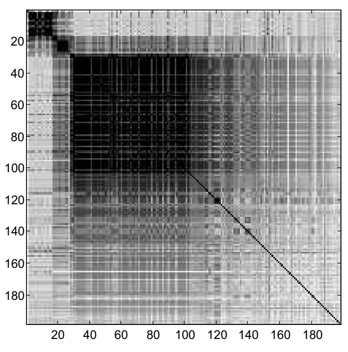
Information about gene products and how they are similar to one another is of great importance in bioinformatics. Traditional approaches use the the DNA sequence as well as the expression values from microarray experiments to produce similarity values. However, additional information, which is more symbolic in nature, is available about gene products in the GO terms and index terms in publications about gene products [Raychaduri and Altman, 2003]. I use these symbolic data about gene products to build visualization and functional summarization.

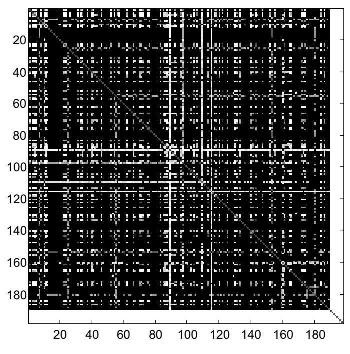
Each gene product, G_i, is represented by a collection of GO terms $G_i = \{T_{i1}, \ldots, T_{in_i}\}$. Previously developed methods of computing similarity measures for two gene products that are annotated by GO terms are described in detail in Keller et al. [2004], Popescu et al. [2006], Keller et al. [2006], Jiang and Conrath [1997], Resnik [1995], Lin [1998], and Lord et al. [2003]. Next, I give a brief overview of these similarity measures along with commentary on the strengths and weaknesses of each.

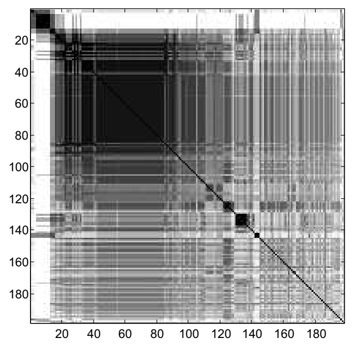

(a) Jaccard

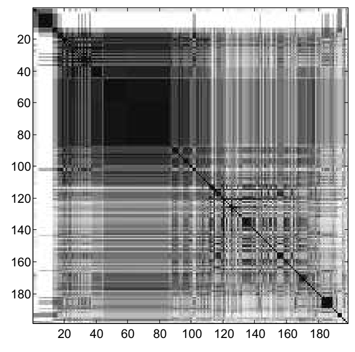

(b) Dice


(c) Cosine


(d) PATH


(e) WUP


(f) Average IC


(g) Normalized Average IC

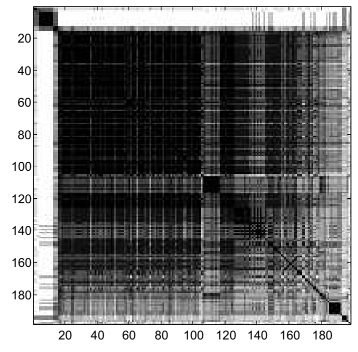

(h) OWA(4) IC

Figure 2.12: VAT-reordered dissimilarity data $\mathbf{D}^*_{TF198}$ of Arabidopsis transcription-factors built with different similarity measures, $\mathbf{D} = 1 - \mathbf{S}$.

Table 2.5: Example GO Annotations for Arabidopsis Transcription Factor-Related Genes, $TF198$.

GO id	GO term	Definition
AT2G35700	GO:0003677	DNA binding
	GO:0003700	transcription factor activity
	GO:0005634	nucleus
	GO:0006355	regulation of transcription, DNA-dep.
AT1G46768	GO:0003677	DNA binding
	GO:0003700	transcription factor activity
	GO:0005634	nucleus
	GO:0006355	regulation of transcription, DNA-dep.
AT4G17710	GO:0003677	DNA binding
	GO:0003700	transcription factor activity
	GO:0005634	nucleus
	GO:0006355	regulation of transcription, DNA-dep.
AT5G60890	GO:0000162	tryptophan biosynthesis
	GO:0003677	DNA binding
	GO:0003700	transcription factor activity
	GO:0005634	nucleus
	GO:0009651	response to salt stress
	GO:0009737	response to abscisic acid stimulus
	GO:0009739	response to gibberellic acid stimulus
	GO:0009751	response to salicylic acid stimulus
	GO:0009753	response to jasmonic acid stimulus
	GO:0009759	indole glucosinolate biosynthesis
	GO:0016301	kinase activity
	GO:0016563	transcriptional activator activity

Set-based and vector-based similarity measures

For two gene products, $G_1 = \{T_{11}, \ldots, T_{1n_1}\}$ and $G_2 = \{T_{21}, \ldots, T_{2n_2}\}$, one can use set-based similarity measures to define a similarity value between G_1 and G_2, denoted $s(G_1, G_2)$. Two such measures are the Jaccard and Dice similarity measures,

$$s_J(G_1, G_2) = \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|}, \tag{2.24}$$

$$s_D(G_1, G_2) = \frac{2|G_1 \cap G_2|}{|G_1| + |G_2|}. \tag{2.25}$$

Figure 2.11(a,b) illustrate these similarity measures for $GPD194_{12.10.03}$.

A property of set-based similarities is that if $G_1 \cap G_2 = \emptyset$, then $s(G_1, G_2) = 0$. This is an undesirable property as two genes that are annotated by GO terms that are near to each other on the hierarchical tree (but not identical) should have a non-zero similarity. In essence, set-based similarities under-represent the similarity values. This problem also is present in vector-based similarity measures, such as the cosine similarity. Vector-based methods are based on a binary vector representation of each gene product, where $\vec{v}_i \in \mathbb{R}^{N_T}$ and N_T is the total number of unique terms in the ontology. Each element of $\vec{v}_i$ has a value of 1 if the gene is annotated by the associated term and 0, otherwise. The cosine similarity is defined as

$$s_{VC}(G_1, G_2) = \frac{\vec{v}_1 \cdot \vec{v}_2}{|\vec{v}_1||\vec{v}_2|}, \tag{2.26}$$

where $\vec{v}_1 \cdot \vec{v}_2$ is the dot product and $|\cdot|$ is the vector norm. Figure 2.11(c) shows D_{194} built with the cosine similarity. Notice that the cosine similarity will produce a non-zero similarity only if the two gene products share a common annotation. Also, N_T is typically very large; thus, vector-based approaches become computationally expensive [Popescu et al., 2006]. The vectors are also quite sparse, which is another bad property of the use of vector-based approaches on GO data.

To address the problems of set-based and vector similarity measures, one can use term-based similarity measures. Pair-wise term similarity values are computed for all pairs of terms between G_1 and G_2 and these pair-wise similarities are aggregated to produce a similarity $s(G_1, G_2)$. Two approaches for computing the similarity between two terms are path-based methods and information-theoretic constructs.

Path-based similarity measures

The most standard form of the path-length similarity measure is the PATH measure which is simply the inverse of the shortest path length between two terms. The shortest path length in an ontology is defined as the path between the two terms and their *nearest-common-ancestor* (NCA).

Two other well-known path-length similarity measures are the LCH [Leacock and Chodorow, 1998] and the WUP [Wu and Palmer, 1994]. The LCH measure normalizes the value of the PATH measure by the maximum path length between all pairs of children of the NCA. Effectively, the LCH combines both the path-length similarity between two terms as well as the specificity of the two terms into a composite measure of similarity. The WUP measure also does this by scaling the shortest-path-length between two terms by the sum of the path-lengths from each term to the root term (the sum of the depths of each term).

Nagar and Al-Mubaid [2008] measure similarity between GO terms by averaging all path-lengths between two terms and using an exponential transfer function to compute similarity. The authors showed that this measure performed comparably to information theoretic measures in a (very) limited empirical study.

Path-based similarity measures are more effective than set-based measures at measuring similarity between ontology terms because they take into account the structure of the tree. Another type of similarity that takes this structure into account is information-theoretic similarity.

Information-theoretic similarity measures

Information-theoretic similarity measures are based on the *information content* of each term. The information content is proportional to the number of occurrences in a corpus of each term and its children from the ontology. Often information content is expressed as the log of probability, such that an information content = 1 indicates that a term appears only once in the corpus, and an information content = 0 indicates that a term or one of its children appear in all entries. This type of representation is very helpful as terms that have a low information content are not as useful for clustering because they are less specific than terms with high information content. Clearly, the corpus used for a specific problem influences the information content value; I use Swiss-Prot-HUMAN for the $GPD194_{12.10.03}$ data — which is composed of human genes — and Swiss-Prot-ARABIDOPSIS for the Arabidopsis examples [The UniProt Consotium, 2007]. There are many types of links in the GO, which can affect how one defines children terms. These most predominant link types are 'is-a' and 'part-of'. I focus my discussion on the numerical techniques for clustering ontological data; hence, I assume that all links are equivalent when computing information content. Tailoring term-based similarity measures to handle different types of links is definitely a pertinent research topic, but is not addressed here.

The probability of term T_k is computed by counting the number of occurrences in the corpus of T_k and all of its children,

$$p(T_k) = \frac{\text{count}(T_k + \text{children}(T_k))}{\text{count}(\text{all terms in corpus})}, 1 \leq k \leq |\text{GO}|. \tag{2.27}$$

Let $\mathbf{T}(T_i, T_j)$ be the set of all possible NCAs of terms T_i and T_j. GO terms can appear as children to more than one parent in the hierarchical taxonomy; hence, the probability is defined as

$$p_{NCA}(T_i, T_j) = \min_{T_k \in \mathbf{T}(T_i, T_j)} p(T_k), \tag{2.28}$$

where $p_{NCA}(T_i, T_j)$ is the probability of the NCA of terms T_i and T_j.

Resnik [1995] defines the pair-wise similarity value of terms T_i and T_j as

$$s_{\mathrm{RES}}(T_i, T_j) = -\log p_{NCA}(T_i, T_j), \tag{2.29}$$

where $0 \leq s_{\mathrm{RES}}(T_i, T_j) \leq -\log(\min_{\forall T} p(T))$. Note that $-\log(p(T)) = \infty$ for terms that do not appear in the corpus; hence, caution must be exercised with this similarity measure (and others based on information content). This similarity measure can be normalized by $-\log(\min_{\forall T} p(T))$, assuming that all terms appear in the corpus. Lin [1998] created a similarity measure that produced values on the interval $[0, 1]$ by normalizing by the information content of T_i and T_j,

$$s_{\mathrm{LIN}}(T_i, T_j) = \frac{2 \log p_{NCA}(T_i, T_j)}{\log p(T_i) + \log p(T_j)}. \tag{2.30}$$

Jiang and Conrath [1997] developed a measure of dissimilarity between terms T_i and T_j that is similar to Lin's similarity measure. However, like Resnik's measure, Jiang's dissimilarity measure is not normalized and produces values on the interval $[0, 2\log(\mathrm{count}_{min})]$,

$$d_{\mathrm{JIANG}}(T_i, T_j) = -2 \log p_{NCA}(T_i, T_j) - (\log p(T_i) + \log p(T_j)). \tag{2.31}$$

Any aggregation method can be used to fuse pair-wise term similarity values—examples include average, maximum, minimum, and *linear combinations of order statistics* (LOS) or *ordered weighted-average* (OWA) operators [Yager and Kacprzyk, 1997]. Consider two gene product GO represntations, $G_1 = \{T_{11}, \ldots, T_{1n_1}\}$ and $G_2 = \{T_{21}, \ldots, T_{2n_2}\}$. The average similarity-measure is computed as

$$s_{AVG}(G_1, G_2) = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} s(T_{1i}, T_{2j}). \tag{2.32}$$

This similarity measure can also be normalized, where

$$s_{NAVG}(G_1, G_2) = \frac{s_{AVG}(G_1, G_2)}{\max\{s_{AVG}(G_1, G_1), s_{AVG}(G_2, G_2)\}}. \tag{2.33}$$

The maximum and minimum similarity measures simply compute the maximum or minimum of the possible pair-wise term similarities between the genes. LOS or OWA operators are shown to be robust to data variability and outliers [Hosking, 1990, Yager and Kacprzyk, 1997] and are calculated as

$$s_{OWA}(G_1, G_2) = \frac{\sum_{i=1}^{n_1 n_2} w_i s_{(i)}}{\sum_{i=1}^{n_1 n_2} w_i}, \tag{2.34}$$

where $s_{(i)}$ is the set of pair-wise term similarities ordered from greatest to least and w_i are the associated weights. For example, if $w_i = 1$, $i \leq 4$, $w_i = 0$, $i > 4$, then this operator could be thought of as the average of "at least 4" pair-wise similarities. The information theoretic similarity measures are useful in the cases where genes do not share a common term. Set-based and vector-based similarity measures will produce a zero-valued similarity; in contrast, the pair-wise term aggregations used in information-theoretic measures will produce a non-zero-valued similarity.

Fuzzy similarity measures

Fuzzy GO similarities were first proposed by Keller et al. [2004]. The *Fuzzy Measure-based Similarity* (FMS) is based on the general fuzzy measure [Sugeno, 1974, 1977] and considers gene product similarity to be a global combination of the sets of GO terms that represent the gene products. The fuzzy measure considers "information sources" to compute fuzzy densities (akin to probability measures) of subsets of the information sources. Let $g : 2^G \to [0, 1]$ be the fuzzy measure, where $g^i = g(\{T_i\})$ is the fuzzy density or importance of GO term T_i in determining the similarity between two genes. GO term information content is a direct measure of a term's importance; hence, $g^i = IC(T_i)$, where $IC(T_i)$ is the normalized information content of T_i.

Restricting the fuzzy measure to a Sugeno λ-measure allows the entire fuzzy measure to be iteratively built from the individual fuzzy densities g^i [Sugeno, 1974, 1977]. A λ-measure has the additional property that for $A, B \subseteq G$ with $A \cap B = \emptyset$,

$$g_\lambda(A \cup B) = g_\lambda(A) + g_\lambda(B) + \lambda g_\lambda(A) g_\lambda(B), \text{ for some for some } \lambda > -1. \tag{2.35}$$

The fuzzy measure for a set of GO terms that define gene product G must satisfy $g_\lambda(G) = 1$. Hence, the value of λ that satisfies this property can be found by solving

$$(1 + \lambda) = \prod_{i=1}^{N_T} (1 + \lambda g^i). \tag{2.36}$$

Sugeno [1974] and Tahani and Keller [1990] showed that this equation has one solution that satisfies $\lambda > -1$. Finally, the FMS is defined as,

$$s_{FMS} = \frac{g_1(G_1 \cap G_2) + g_2(G_1 \cap G_2)}{2}, \tag{2.37}$$

where g_1 is the Sugeno λ-measure defined on G_1 and g_2 is defined on G_2. The following example from Popescu et al. [2006] outlines the FMS calculation for two genes in $GPD194_{12.10.03}$.

Example 2.4.1 (Fuzzy similarity measure). Consider two gene products, G_1 is GenBank ID AAH35609 (MTMR4 gene) and G_2 is GenBank ID AAH12399 (MTMR8 gene). The GO terms for these gene products are

$$G_1 = \{T_1 = \text{GO} : 0004721, T_2 = \text{GO} : 0006470, T_3 = \text{GO} : 0008270\},$$

and

$$G_2 = \{T_1 = \mathrm{GO}:0004721, T_2 = \mathrm{GO}:0006470, T_4 = \mathrm{GO}:0016787\}.$$

The fuzzy densities (normalized information contents) of these terms are $\{g^{1i}\} = \{0.52, 0.57, 0.54\}$ and $\{g^{2i}\} = \{0.52, 0.57, 0.33\}$. The set of common terms $G_1 \cap G_2 = \{T_1, T_2\}$, which is the set of terms that defines the similarity of G_1 and G_2. The λ-measure on G_1 has $\lambda = -0.84$, from Eq.(2.36). The λ-measure on G_2 has $\lambda = -0.72$. Applying Eq.(2.35), $g_1(\{T_1, T_2\})$ and $g_2(\{T_1, T_2\})$ are calculated as,

$$\begin{aligned} g_1(\{T_1, T_2\}) &= g^1 + g^2 + \lambda g^1 g^2 \\ &= 0.52 + 0.57 - 0.84 * 0.52 * 0.57 \\ &= 0.84 \\ g_2(\{T_1, T_2\}) &= 0.88 \end{aligned}$$

The FMS similarity of G_1 and G_2 is

$$s_{FMS}(G_1, G_2) = \frac{0.84 + 0.88}{2} = 0.86.$$

Note that G_1 and G_2 are in the same protein family, myotublarin; hence, a high similarity would be expected.

The FMS shares the same problem as set-based similarities as genes that do not share common terms have zero similarity under the FMS. Thus, the *Augmented Fuzzy Measure-based Similarity* (AFMS) was proposed [Keller et al., 2004]. The AFMS alleviates the problem of the FMS by augmenting G_1 and G_2 by the NCAs of all the pairs of terms in G_1 and G_2, where

$$G_1' = G_1 \cup \{T_{1i,2j}\},\ G_2' = G_2 \cup \{T_{1i,2j}\}.$$

The set $\{T_{1i,2j}\}$ is the set of all NCAs of each pair of terms (T_{1i}, T_{2j}), where T_{1i} are the individual terms in G_1 and T_{2j} are the individual terms in G_1. The intersection $\{G_1 \cap G_2\}$ in Eq.(2.37) is replaced by the intersection of the augmented sets $\{G_1' \cap G_2'\}$, producing

$$S_{AFMS}(G_1, G_2) = \frac{g_1(G_1' \cap G_2') + g_2(G_1' \cap G_2')}{2}. \tag{2.38}$$

Note that the AFMS similarity between two genes will always be non-zero as the augmented sets will always share the NCA terms[1].

Keller et al. [2004] also present a Choquet-integral based similarity-measure. However, this measure requires an additional piece of information, the expected worth or evidence weight of each annotating term. The authors map the GO evidence codes, which consist of linguistic identifiers such as *inferred from experiment* (EXP), *traceable author statement* (TAS), and *non-traceable author statement* (NAS) to a set of numerical weights. These weights are then used to produce a Choquet-based similarity-measure. This measure is effective for computing similarity between genes and gene products, but I do not specifically address it in this dissertation because it is dependent on the evidence codes.

2.4.2 Cluster Summarization

In general practice, cluster summarization is the act of creating a feature-summarization of the objects within a single cluster. The objects in a cluster are summarized according to their shared attributes or the attributes that are most common among them.

A specific form of cluster summarization that is very pertinent to biology is the

[1] There is one exception to this rule. If the set of terms $\{T_{1i,2j}\}$, the NCAs of the terms from the two genes, only contains the root node in the GO (or a set of the root nodes, if the three GO trees are treated independently), then the AFMS will be 0.

functional summarization of genes and gene products. This is an important task in bioinformatics. Often researchers wish to determine the function of a gene and to do this they infer the function by finding genes with known function that are similar (such as in sequence or GO annotation). The GO is a powerful tool in this exploit because, not only can gene similarity be measured, but the GO terms themselves are functional annotations.

There are many algorithms for summarizing genes and gene products. Kankar and Mukherjea [2005] created a summarization of a list of genes by looking for common MeSH [MeS] terms among PubMed [Pub] queries. Hu [2004] used ensemble methods to combine expression analysis and text summarization to compute summarizing keywords of groups of genes. Finally, Popescu et al. [2004] provide a method by which summarizations are computed using the results of fuzzy clustering. The partition membership values are used to compute weighted frequency measures of the GO terms annotating the gene products. Gene products that have a high membership in a cluster have more weight in the summarization calculation.

In essence, each of these cluster summarization methods use a counting method to compute the most common features among a group of objects. The common features are the summary of the function (or attributes) of the objects.

2.4.3 Self Organizing Maps (SOM)

The self-organizing map is a two-layer lateral feedback neural network that topologically maps itself to the training data. Like methods such as *principle-component-analysis* (PCA) [Hotelling, 1933] and *multidimensional-scaling* (MDS) [Kruskal and Wish, 1978], the SOM allows one to visualize the cluster tendency of multi-dimensional data. The network structure is often set to a two-dimensional square, toroidal, or hexagonal grid, where each network node, or prototype, is laterally connected to its

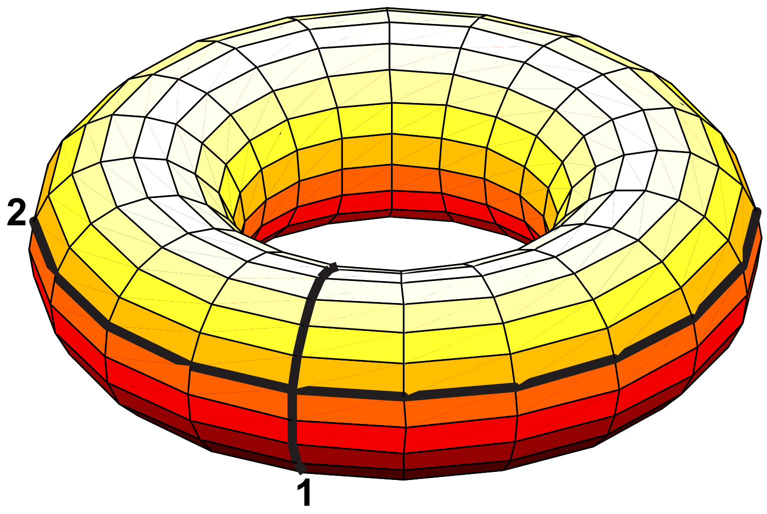


Figure 2.13: 20 × 20 toroidal grid SOM network.

neighbors. Figure 2.13 shows an example of a toroid network structure. This network can be visualized in 2-dimensions by forming the network into a cylinder by cutting along the dark line denoted ‘1’ and then cutting the cylinder along the dark line denoted ‘2’ to form a plane.

Algorithm 2.4.1: Self-Organizing Map [Kohonen, 1981]

```
Data: Training data $\mathbf{X} = \{\vec{x}_1, \ldots, \vec{x}_N\}$
Randomly initialize prototype vectors, $\vec{w}_i \in \mathbb{R}^D, \forall i$.
for $t = 1$ to $t_{max}$ do
    Randomly draw a test signal, $\vec{x}_d$.
    $p = \arg\min_i\{||\vec{x}_d - \vec{w}_i||\}$
    $\vec{w}_i = \vec{w}_i + \epsilon(t) \cdot h_{ip}(t) \cdot (\vec{x}_d - \vec{w}_i), \forall i$
    $\sigma(t) = \sigma_0(\sigma_f/\sigma_0)^{t/t_{max}})$
    $\epsilon(t) = \epsilon_0(\epsilon_f/\epsilon_0)^{t/t_{max}})$
    $t \leftarrow t + 1$
```

Algorithm 2.4.1 outlines the SOM algorithm. Essentially, the learning procedure draws random test signals from the training data and moves the prototypes towards this test signal according to a neighborhood influence function h_{ip}. This function determines the influence of each update on the network, where winning prototype (the one closest to the test signal) are moved more than its neighbors. Often the neighborhood influence function h_{ip} is Gaussian, defined as

$$h_{ip}(t) = \exp\left(-\frac{|\vec{a}_i - \vec{a}_p|^2}{\sigma^2(t)}\right), \tag{2.39}$$

where $\vec{a}_i$ is the location of the SOM prototype in the predefined neighborhood (e.g. square or hexagonal grid) and $\sigma^2(t)$ is the width of the influence function. The SOM training procedure is repeated until a maximum number of iterations (t_{max}) or convergence is reached. Typically, the learning rate $\epsilon(t)$ and the width of the neighborhood function $\sigma^2(t)$ are reduced during iteration, with the effect that late iterations are only applying small updates to network prototypes local to the winning prototype p. Kohonen [2001] described the process by which the prototypes align themselves to the input data as *ordering*. In the early iterations, when the learning rate σ and width of the influence function ϵ are large, the network is aligning itself roughly to the distribution of the input data. In late iterations the network does not change significantly but *refines* itself to the substructure of the input data. He showed that the weight vectors order themselves according to the topology of the network and the distribution of the input data such that the distance between the prototypes accurately models the distance between the input data samples.

There are many variants of the SOM: over 7,500 papers are identified in the bibliographies of Kaski et al. [1998], Oja et al. [2003], and Pöllä et al. [2006]. SOM variants of note include growing SOMs [Fritzke, 1991], neural gas [Martinez et al., 1993], and temporal varieties [Hammer et al., 2004]. However, the SOM implicitly is ill-suited for ontological data or data that cannot be represented by object vectors.

Self Organizing Semantic Map (SOSM) and WEBSOM

Since Kohonen developed the SOM in 1981 [Kohonen, 1981], the SOM has been adapted to many types of data, including document data. In 1989, Ritter and Kohonen developed the *Self-Organizing Semantic Map* (SOSM) [Ritter and Kohonen, 1989], which represents the semantic features of objects by a binary-valued vector. However, the SOSM does not incorporate the similarities between the semantic features. Hence, correlated semantic dimensions are considered to be independent.

The WEBSOM algorithm [Konkela et al., 1996] addresses this weakness of the SOSM by first computing a *word category map* and then using this to create a *document map*. The similarities of words or terms are encoded in the word category map by imputing similarity from their relative placement and frequency in documents. WEBSOM, however, is ill-suited to the ontological data in the GO or MeSH, which are both represented by a *directed-acyclic-graph* (DAG).

Relational Batch SOM

Recently, Hasenfuss and Hammer [2007] developed a relational variant of the SOM by extending the relational duals of c-means clustering algorithms [Hathaway et al., 1989] to topographic maps. Hence, one can compute the SOM of ontological data by using the relational SOM on the dissimilarity matrix of the objects: genes, documents, etc. The prototypes in the relational batch SOM, denoted $\vec{\alpha}$, represent the weight of each object in defining the prototype. The standard SOM prototype $\vec{w}$ is related to $\vec{\alpha}$ by

$$\vec{w} = \sum_i (\vec{\alpha})_i \vec{x}_i, \tag{2.40}$$

where $\vec{x}_i$ is the ith training data vector and $\vec{\alpha}$ is constrained to $|\vec{\alpha}| = 1$. Essentially, the prototypes live in the convex-hull of the training data $\mathbf{X}$. The distance between the prototype $\vec{\alpha}$ and the test signal $\vec{x}_i$ is computed as it is in the relational duals of c-means clustering algorithms [Hathaway et al., 1989],

$$d_R^2(\vec{\alpha}_i, \vec{x}_j) = ||\vec{w}_i - \vec{x}_j||^2 = (\mathbf{D}\vec{\alpha}_i)_j - 0.5\vec{\alpha}_i\mathbf{D}\vec{\alpha}_i, \tag{2.41}$$

where $D_{kl} = ||\vec{x}_k - \vec{x}_l||$ is the pairwise dissimilarity matrix and $(\mathbf{D}\vec{\alpha}_i)_j$ is the jth element of the matrix-vector multiplication. Algorithm 2.4.2 outlines the relational batch SOM.

Algorithm 2.4.2: Relational Batch SOM [Hasenfuss and Hammer, 2007]

```
Data: Training data $\mathbf{X} = \{\vec{x}_1, \ldots, \vec{x}_N\}$
Randomly initialize prototype vectors, $\vec{\alpha}_i \in [0,1]^N, \forall i$.
$\vec{\alpha}_i = \frac{\vec{\alpha}_i}{|\vec{\alpha}_i|}, \forall i$
$t \leftarrow 0$
while $t < t_{max}$ do
    $\vec{\alpha}'_i = \{0\}^N, \forall i$
    for $j = 1$ to $N$ do
        Find closest prototype, $p = \arg\min_i d_R^2(\vec{\alpha}_i, \vec{x}_j)$.
        $\alpha'_{ij} = \alpha'_{ij} + h_{ip}(t), \forall i$
    $\vec{\alpha}_i = \frac{\vec{\alpha}'_i}{|\vec{\alpha}'_i|}, \forall i$
    $\sigma(t) = \sigma_0(\sigma_f/\sigma_0)^{t/t_{max}}$
    $t \leftarrow t + 1$
```

The drawback of using this method is that the prototypes do not directly encode the ontological data; hence, summarization is not as straight forward as with the proposed OSOM method. Additionally, one must choose a pairwise gene similarity measure, which further reduces the ability to generalize the SOM to ontological data.

This chapter summarized methods by which one can analyze relational data. The next chapters will introduce theoretical analysis on algorithms for relational data and new ways of grouping objects and visualizing the results.

Chapter 3

Analysis

3.1 Aligned Partitions

As was shown in Section 2.1, there are visualization techniques, such as VAT, that address the cluster tendency question by displaying the dissimilarity data such that the clusters are seen as blocks along the diagonal. Assume that the dissimilarity data has been reordered by some algorithm to produce a "VAT-like" image, as shown in Fig. 2.4. The important property of $I(\mathbf{D}^*)$ is that it has, beginning in the upper left corner, dark blocks along its main diagonal. Under this assumption, the set of crisp c-partitions that mimic the blocky structure in $I(\mathbf{D}^*)$ is a proper subset of the set of all crisp c-partitions, $\mathbf{M}_{hcn}$. I call these partitions, $\mathbf{U} \subsetneq \mathbf{M}_{hcn}$, *aligned* partitions. Aligned c-partitions of $\mathbf{O}$ have c contiguous blocks of 1's in $\mathbf{U}$, ordered to begin with the upper left corner and proceeding down and to the right. The set of all aligned c-partitions is

$$\mathbf{M}^*_{hcn} = \{\mathbf{U} \in \mathbf{M}_{hcn} | u_{1k} = 1, 1 \leq k \leq n_1 : u_{ik} = 1, n_{i-1} + 1 \leq k \leq n_i, 2 \leq i \leq c\}. \tag{3.1}$$

For example, $\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$ are aligned partitions,

while $\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$, and $\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$ are not.

The special nature of aligned partitions enables us to specify them in an alternative form. Every member of $\mathbf{M}^*_{hcn}$ is isomorphic to the unique set of c distinct integers (which are the cardinalities of the c clusters in $\mathbf{U}$) that satisfy
$\{n_i | 1 \le n_i; 1 \le i \le c; \sum_{i=1}^{c} n_i = n\}$. So *aligned* paritions are completely specified by $\{n_1 : \ldots : n_c\}$. For example,

$$\mathbf{U} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \{2 : 1 : 2\}. \tag{3.2}$$

As stated before, the set of aligned c-partitions is a proper subset of the set of all crisp c-partitions, $\mathbf{M}^*_{hcn} \subsetneq \mathbf{M}_{hcn}$. But what is the difference between the cardinality of $\mathbf{M}^*_{hcn}$ and $\mathbf{M}_{hcn}$? This is an important question because, as shown later, CLODD extracts aligned partitions from reordered dissimilarity data. The following proposition provides an answer.

Proposition 3.1.1. *The cardinality of $\mathbf{M}^*_{hcn}$, the set of aligned c-partitions of n objects into $2 \le c < n$ crisp subsets in $\mathbf{M}_{hcn}$, is*

$$|\mathbf{M}^*_{hcn}| = \begin{pmatrix} n-1 \\ c-1 \end{pmatrix} \tag{3.3}$$

Proof. Recall that aligned partitions can be completely specified by $\{n_1 : \ldots : n_c\}$. Hence, the cardinality of $\mathbf{M}^*_{hcn}$ is equal to the cardinality of $\{n_1 : \ldots : n_c\}$, under the constraints

$$n_i \in \mathbb{Z}; 1 \leq n_i \leq (n-c+1) \forall i; \sum_{i=1}^{c} n_i = n. \tag{3.4}$$

Consider n_i to be the number of marbles in a bag or container, where there are c bags. You are given n marbles to put in those bags under the constraint that you must place at least one marble in each bag and you cannot be left with any marbles. How many different ways could you place the marbles in the bags? Solving this problem is equivalent to proving Proposition 3.1.1.

Begin by placing one marble in each bag. There are $(n-c)$ marbles left over. Hence, the maximum number of marbles that could be in any one bag is $(n-c+1)$. Now, choose a bag at random and add one marble to its contents. Continue until all marbles are placed. Thus, you have c objects (bags) to choose from and you choose $(n-c)$ times. The order does not matter and the objects (bags) can be chosen more than once. Thus, this is a well known combinatorics problem, where one chooses an unordered sample of size $(n-c)$ with repetition from a population of c elements [Epp, 2004]. The number of combinations is the value of the binomial coefficient,

$$\begin{pmatrix} c+(n-c)-1 \\ c-1 \end{pmatrix} = \begin{pmatrix} n-1 \\ c-1 \end{pmatrix}, \tag{3.5}$$

which is Eq.(3.3). □

Remark 3.1.1. For $c << n$, $|\mathbf{M}^*_{hcn}| \approx n^{c-1}/(c-1)!$ is a good approximation to the exact value in Eq.(3.3). The exact cardinality of $\mathbf{M}_{hcn}$ is known, $|\mathbf{M}_{hcn}| = \frac{1}{c!}\sum_{j=1}^{c} \begin{pmatrix} c \\ j \end{pmatrix} (-1)^{c-j} j^n$. For $c << n$, the last term dominates this sum, and the approximation $|\mathbf{M}_{hcn}| \approx c^n/c!$ can be used. It is instructive to compare the size of

$\mathbf{M}^*_{hcn}$ to that of $\mathbf{M}_{hcn}$, by the ratio

$$\frac{|\mathbf{M}^*_{hcn}|}{|\mathbf{M}_{hcn}|} \approx \frac{n^{c-1}/(c-1)!}{c^n/c!} = \left(\frac{n^{c-1}}{(c-1)!}\right)\left(\frac{c!}{c^n}\right) = \frac{n^{c-1}}{c^{n-1}},\ c << n. \tag{3.6}$$

Applying this ratio for the fairly typical problem of $c = 10$ and $n = 10,000$ yields $|\mathbf{M}^*_{hcn}|/|\mathbf{M}_{hcn}| \approx 1/10^{9963}$ — a *very* small number. This shows that algorithms that search for a crisp partition of $\mathbf{D}$ over $\mathbf{M}^*_{hcn}$ have a significantly smaller set of solutions to examine. Note, however, that the size of $\mathbf{M}^*_{hcn}$ is still quite large: for $c = 10$ and $n = 10,000$, $|\mathbf{M}^*_{hcn}| \approx 10^{36}/9! = 2.7557 \times 10^{30}$. Hence, even though $\mathbf{M}^*_{hcn}$ is relatively small, it is still far too big for exhaustive search. This is further discussed in Section 4.1.

The transformation $\mathbf{U}^T\mathbf{U}$ produces an $n \times n$ matrix with elements that are a measure of the relationship between pairs of objects in each of c clusters Huband and Bezdek [2008]. The ijth element of $\mathbf{U}^T\mathbf{U}$ is $\left(\mathbf{U}^T\mathbf{U}\right)_{ij} = \sum_{k=1}^{c} u_{ki}u_{kj}$; thus, $\left(\mathbf{U}^T\mathbf{U}\right)_{ij}$ is a measure of the binding between objects i and j over all c clusters. This dissertation primarily addresses crisp partitions $\mathbf{U} \in \mathbf{M}_{hcn}$, where all elements are 1 or 0; hence, $\left(\mathbf{U}^T\mathbf{U}\right)_{ij}$ will be zero unless objects i and j are in the same cluster k, in which case the ijth entry will have the value 1.

A property of aligned c-partitions is that the image of the transformation $([1] - (\mathbf{U}^*)^T\mathbf{U}^*)$ has dark blocks along its diagonal: black represents 0 in the image, white represents 1. This image of the transformation is denoted as $T(\mathbf{U}^*) = ([1] - (\mathbf{U}^*)^T\mathbf{U}^*)$, where $[1]$ is the $n \times n$ matrix of 1's. The following example shows the transformation for $c = 2$, $n = 5$, and $\mathbf{U}^* = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$;

$$(\mathbf{U}^*)^T \mathbf{U}^* = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{3.7}$$

This example shows that the blocks of 1's correspond to the grouping of the objects according to the partition. Figure 3.4(d) shows an image of $T(\mathbf{U}^*)$ for the visually-apparent aligned 3-partition of objects arranged as three parallel lines.

3.2 VAT and Single-Linkage

Consider a dissimilarity matrix $\mathbf{D}$, which is then reordered by VAT with Prim's MST algorithm. The VAT ordering of the MST is a special subset of all possible orderings computed by Prim's algorithm, where VAT imposes the following property by its initialization, (see Eq.(2.1) in Algorithm 2.1.1):

$$D^*_{1n} = D^*_{n1} = \max_{i,j} D_{ij}.$$

The discussion in this section is generalized by including all possible orderings of the MST that result from Prim's algorithm. $\mathbf{O}^* = \{o^*_1, \ldots, o^*_n\}$ represents the set of reordered objects as a result of Prim's algorithm (where ANY object can be the starting object in Prim's algorithm). Similarly, $\mathbf{D}^*$ corresponds to the Prim's algorithm ordering $\mathbf{O}^*$. Because the VAT ordering is a subset of the orderings imposed by Prim's algorithm, the discussion in this section also applies to VAT.

Let the objects in any reordered object data $\mathbf{O}^*$ represent the vertices of a fully connected graph, where the individual vertices are denoted as o_i^*, $i = 1, \ldots, n$. The edge weights of this graph are computed by a chosen vector norm or dissimilarity measure d,

$$\text{edge weight}_{ij} = d(o_i^*, o_j^*). \tag{3.8}$$

Subsequently, the total strength of the i-th MST edge, denoted as e_i, has a weight that can be computed by

$$w_i = d_{SL}(\{o_1^*, \ldots, o_i^*\}, \{o_{i+1}^*\}),\ i = 1, \ldots, n-1. \tag{3.9}$$

This is a direct result of Prim's algorithm, which adds edges to the MST by finding the closest safe vertex to the set of vertices previously added to the tree. Notice that in the context of graph theory vertices (objects) are separated into c subtrees, where in the context of clustering this corresponds to a c-partition of the (corresponding) objects.

3.2.1 Aligned Partitions in VAT and SL

Now, consider the relation between SL clustering and the MST. As stated in Section 2.2.2, SL clustering can be performed by cutting the MST. SL divisive clustering cuts the MST at the highest weight edge(s) and the resulting subtrees are the SL clusters (for a given choice of c [the number of clusters]). Furthermore, if the $n(n-1)/2$ off-diagonal dissimilarity values are distinct, the MST will be unique and the same for all similarity measures that are monotonic [Gower and Ross, 1969].

Two lemmas are needed in order to prove that the SL clusters at every c derived by applying Prim's algorithm to any $\mathbf{D}$ with $n(n-1)/2$ distinct off-diagonal dissimilarities are aligned c-partitions of $\mathbf{O}^*$.

Lemma 3.2.1. *The edge weight w_i of edge e_i in an MST with distinct edge weights satisfies*

$$w_i < d_{SL}(\{o_1^*, \ldots, o_i^*\}, \{o_{i+2}^*, \ldots, o_n^*\}),$$

for reordered object data with unique dissimilarity values.

Proof. First, the dissimilarity values are unique; thus,

$$d_{SL}(\{o_1^*, \ldots, o_i^*\}, \{o_{i+2}^*, \ldots, o_n^*\}) \neq w_i.$$

Second, Prim's algorithm states that the $(i+1)$-th vertex added to the MST is the vertex closest to the subtree $\{o_1^*, \ldots, o_i^*\}$, namely o_{i+1}^*. Hence, all edges that connect the subtrees $\{o_1^*, \ldots, o_i^*\}$ and $\{o_{i+2}^*, \ldots, o_n^*\}$ must be greater in weight than edge e_i, resulting in the identity

$$d_{SL}(\{o_1^*, \ldots, o_i^*\}, \{o_{i+2}^*, \ldots, o_n^*\}) > w_i,$$

which proves the lemma. □

Next, Lemma 3.2.1 is used to prove

Lemma 3.2.2. *If $w_m = \max_j w_j$, then cutting the associated MST edge e_m produces the two subtrees, $\{o_1^*, \ldots, o_m^*\}$ and $\{o_{m+1}^*, \ldots, o_n^*\}$.*

Proof. Lemma 3.2.1 shows that

$$w_m < d_{SL}(\{o_1^*, \ldots, o_m^*\}, \{o_{m+2}^*, \ldots, o_n^*\}). \tag{3.10}$$

Also, because e_m is the highest weight edge it follows that

$$w_m > w_i \, \forall i \neq m. \tag{3.11}$$

Hence, if there exists an MST edge, other than e_m, that connects the two subtrees $\{o^*_1, \dots, o^*_m\}$ and $\{o^*_{m+1}, \dots, o^*_n\}$, this edge must satisfy both Eq.(3.10) and Eq.(3.11). These two conditions cannot be met simultaneously. Thus, there is no edge in the MST that connects the two subtrees, $\{o^*_1, \dots, o^*_m\}$ and $\{o^*_{m+1}, \dots, o^*_n\}$, if e_m is cut. $\square$

Please note that the subtrees that result from cutting the MST at its maximum weight edge are analogous to an aligned 2-partition of the ordered object data $\mathbf{O}^*$. Now I am ready to state and prove the main result by applying lemma 3.2.2 to divisive clustering.

Proposition 3.2.1. *For a set of object data for which unique dissimilarity values can be computed (or chosen) from a monotonic dissimilarity measure, the SL clusters will be aligned c-partitions of the Prim's algorithm reordered object data* $\mathbf{O}^*$ *for every value of c.*

Proof. Sort the edges of the MST according to their weight in decreasing order, where $e_{(1)} > e_{(2)} > \dots > e_{(n-1)}$ are the sorted edges. The $(c = 2)$ SL clusters can be computed by cutting the MST at the highest weighted edge $e_m = e_{(1)}$. Lemma 3.2.2 proves that the resulting subtrees correspond to the aligned 2-partition, $\{o^*_1, \dots, o^*_m\}$ and $\{o^*_{m+1}, \dots, o^*_n\}$. Furthermore, $\{e_1, \dots, e_{m-1}\}$ are the MST edges (in Prim's algorithm order) of the subtree $\{o^*_1, \dots, o^*_m\}$, and $\{e_{m+1}, \dots, e_{n-1}\}$ are the MST edges (in Prim's algorithm order) of the subtree $\{o^*_{m+1}, \dots, o^*_n\}$. The $(c = 3)$ SL clusters are found by cutting the subtrees at edge $e_{(2)}$. Note that this results in only one of the subtrees being cut, which occurs at its maximum weight edge. Hence, recursive application of lemma 3.2.2 for each of the $n - 1$ steps from $c = n$ to $c = 1$ shows that the resulting subtrees of this cut will also represent aligned partitions. Thus, each SL c-partition of the Prim's algorithm ordered object data is aligned. $\square$

Remark 3.2.1. Proposition 3.2.1 only applies to relational data that have a unique

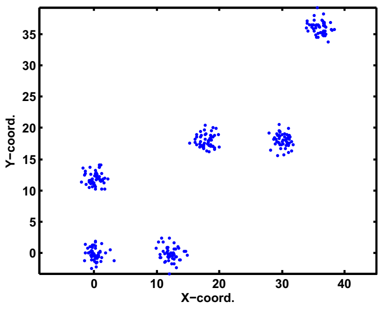
MST (which includes most real-world data sets). However, if needed, a small perturbation of *any* data set (i.e., adding a small random noise to each element in the data) will transform it into one that does have a unique MST. Furthermore, these small perturbations will not affect the cluster structure in the data as they are negligible, in practice. Hence, the analysis, thus far, does apply to most, if not all, real-world data sets.

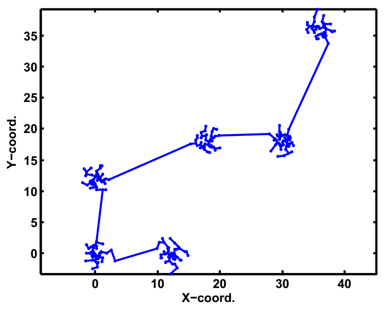
3.2.2 Numerical Examples

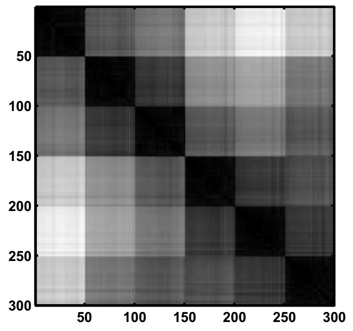
This section contains three examples that illustrate the relationship of SL and VAT explained in the previous section. The dissimilarity data is computed from numerical data by the Euclidean norm for all examples, except the Bioinformatics example (which is a pure relational data set). In order to quantize the *compact, separated* (CS) property of the visually apparent partition in the first two examples, I use *Dunn's validity index* [Dunn, 1974]. Dunn's index is described in detail in Section 3.3, but is essentially a measure of the separation of clusters as compared to their size. I use Dunn's CS index to describe the CS property of the clusters in examples 3.2.1 and 3.2.2

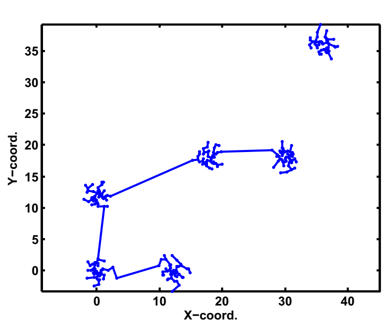
Example 3.2.1 (Six Gaussian clouds). This example will show how SL and VAT detect two-dimensional symmetrical Gaussian-distributed clusters. Table 3.1 shows the statistics of each of the six Gaussian-distributed clouds shown in Fig. 3.1(a). Note the covariance of each cloud is $\sigma^2\mathbf{I}_2$, where $\mathbf{I}_2$ is the 2×2 identity matrix. Dunn's index for the visually apparent crisp 6-partition of this data, shown in Fig. 3.1(h), is 1.2; so, by Dunn's definition, this data contain six CS clusters. To contrast, Dunn's index for the visually apparent 3-partition, shown in Fig. 3.1(f) is 0.7, which indicates that the 3-partition is not CS clusters.

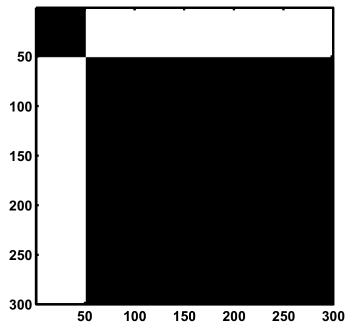
Figure 3.1(b) shows the MST for this data set and Fig. 3.1(c) shows the VAT

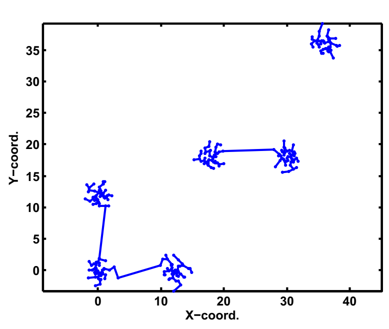

(a) Numerical object data - $\mathbf{X}$

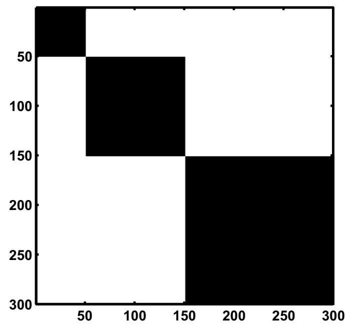

(b) Minimal spanning tree

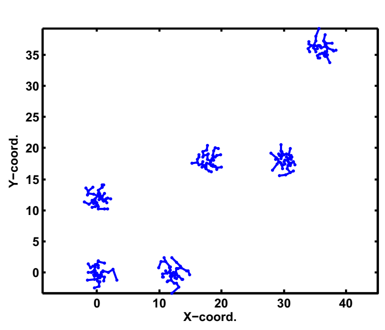

(c) VAT image - $I(\mathbf{D}^*)$


(d) $c = 2$ subtrees


(e) $T(\mathbf{U}_{SL}^{(2)*})$


(f) $c = 3$ subtrees


(g) $T(\mathbf{U}_{SL}^{(3)*})$


(h) $c = 6$ subtrees

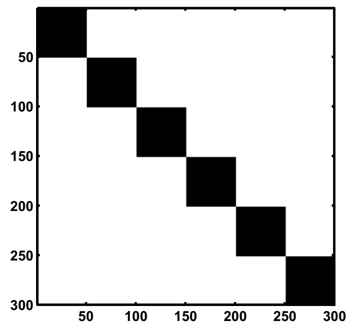

(i) $T(\mathbf{U}_{SL}^{(6)*})$

Figure 3.1: Six Gaussian clouds example: Images of the partition transformations $T(\mathbf{U}_{SL}^*)$ show that the SL partitions are aligned in the VAT ordered data.

Table 3.1: Six Gaussian Clouds Data Characteristics.

No. of objects	Mean, μ	Var., $\sigma^2\mathbf{I}_2$	Indices in VAT
50	(0,0)	1	201-250
50	(12,0)	1	251-300
50	(0,12)	1	151-200
50	(18,18)	1	101-150
50	(30,18)	1	51-100
50	(36,36)	1	1-50

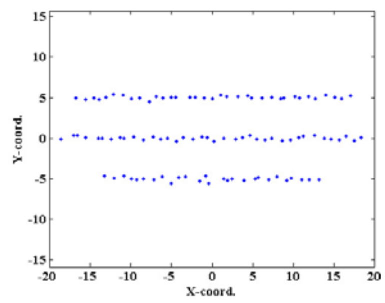
image. The VAT image clearly shows six dark blocks on the diagonal, indicating a tendency for six clusters. Figs. 3.1(d-i) illustrate the results of SL clustering for $c = 2$ clusters, $c = 3$ clusters, and $c = 6$ clusters. In order to show that each partition of the VAT-reordered data is aligned, the image of the partition transformation $T(\mathbf{U}_{SL}^{(c)*})$ is shown, where c is the number of clusters, SL indicates SL, and * indicates that the partition is of VAT-reordered objects. These figures clearly show that the SL partitions at $c = 2$, $c = 3$, and $c = 6$ of the VAT-reordered objects are aligned (as they must be according to Proposition 3.2.1); the clusters appear as c contiguous dark blocks along the diagonal of the image of $T(\mathbf{U}_{SL}^{(c)*})$.

Example 3.2.2 (Three lines). This data set consists of 100 numerical objects arranged as three "parallel lines". Figure 3.2(a) illustrates this data set. Clearly, there are three clusters in the form of long strings of closely-spaced objects. However, Dunn's index for the visually appealing 3-partition of this data (i.e., the parition that groups together all the points along each line) is just 0.12. Consequently, the visually apparent clusters are not CS clusters. This does not, of course, preclude the possibility that the three lines data have some as yet undiscovered c-partition for which Dunn's index is greater than 1, but it is hard to imagine that such could be the case. However, these data do display a strength of SL clustering: the ability to label long stringy groups of objects as belonging to the same cluster. Contrastively, this is a case where VAT fails to indicate the tendency to three clusters. Figure 3.2(c) shows the VAT image of these data. I leave it to you, the reader, to determine how many clusters you see in the VAT image—it surely is not three!

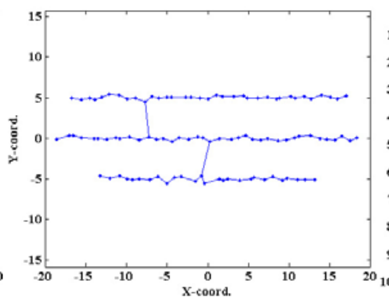
The resulting subtrees of the numerical object data are shown in Figs. 3.2(d,f,h) as a result of SL clustering and the respective images of the transformation $T(\mathbf{U}_{SL}^{(c)*})$ in Figs. 3.2(e,g,i). Notice that the $c = 2$ and $c = 3$ cases show that the VAT image has definite edges at the cut locations. In addition, despite the fact that $c = 6$ clusters is obviously a poor choice for this data set, the $c = 6$ partition of the VAT-reordered data is still aligned. This is the case for *all* choices of c, where $1 \leq c \leq 100$ for this data set.

This example brings up an interesting question regarding the efficacy of VAT to show the cluster tendency of data such as this—why is there such a dichotomy between the strength of SL with long stringy clusters and the weakness of VAT to show the tendency of these clusters? By applying the usual validation heuristic—choose the partition that is before the biggest jump in the cut edge strength—to the SL hierarchy, it selects the visually preferable 3-partition of $\mathbf{D}$. Fig. 3.2(f) makes it clear that SL finds this solution. This issue is addressed in more detail in Section 3.3.

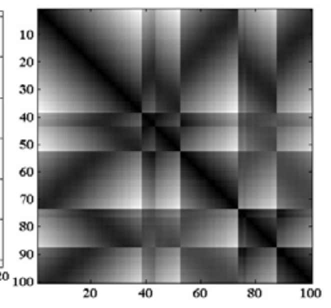
Example 3.2.3 (Bioinformatics data). This example is with a real world data set derived from $GPD194_{12.10.03}$ (characterized in Table 2.4). The dissimilarity data is denoted here as $\mathbf{D}_{194}$. These data are different from those of the previous two examples in that those are derived from object data, while these data are derived directly from a (dis)similarity relation built with a fuzzy measure applied to GO annotations of the $GPD194_{12.10.03}$ genes. Furthermore, the off-diagonal elements of the dissimilarity matrix are not distinct. However, you will see that the analysis still holds true. Section 2.4 contains a detailed description of the construction of this data. These data, $\mathbf{D}_{194}$, are displayed in Fig. 3.3(a) and the VAT image is shown in Fig. 3.3(b). The three ENSEMBL families are clearly visible in both views.



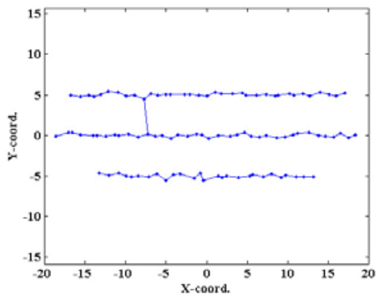
(a) Numerical object data - **X**



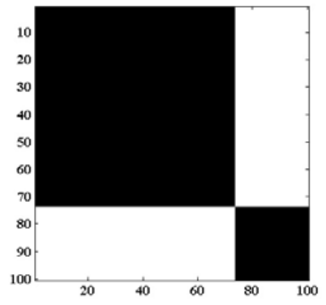
(b) Minimal spanning tree



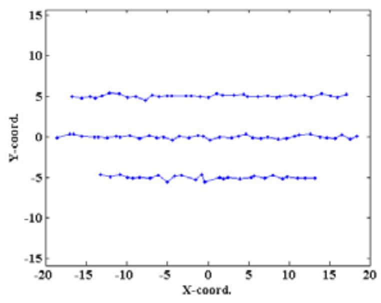
(c) VAT image - $I(\mathbf{D}^*)$



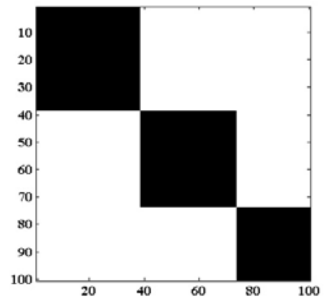
(d) $c = 2$ subtrees



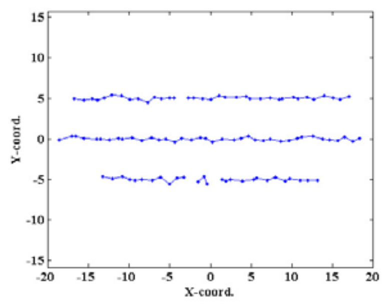
(e) $T(\mathbf{U}_{SL}^{(2)*})$



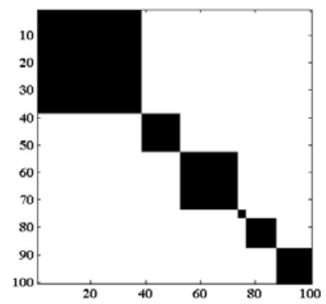
(f) $c = 3$ subtrees



(g) $T(\mathbf{U}_{SL}^{(3)*})$

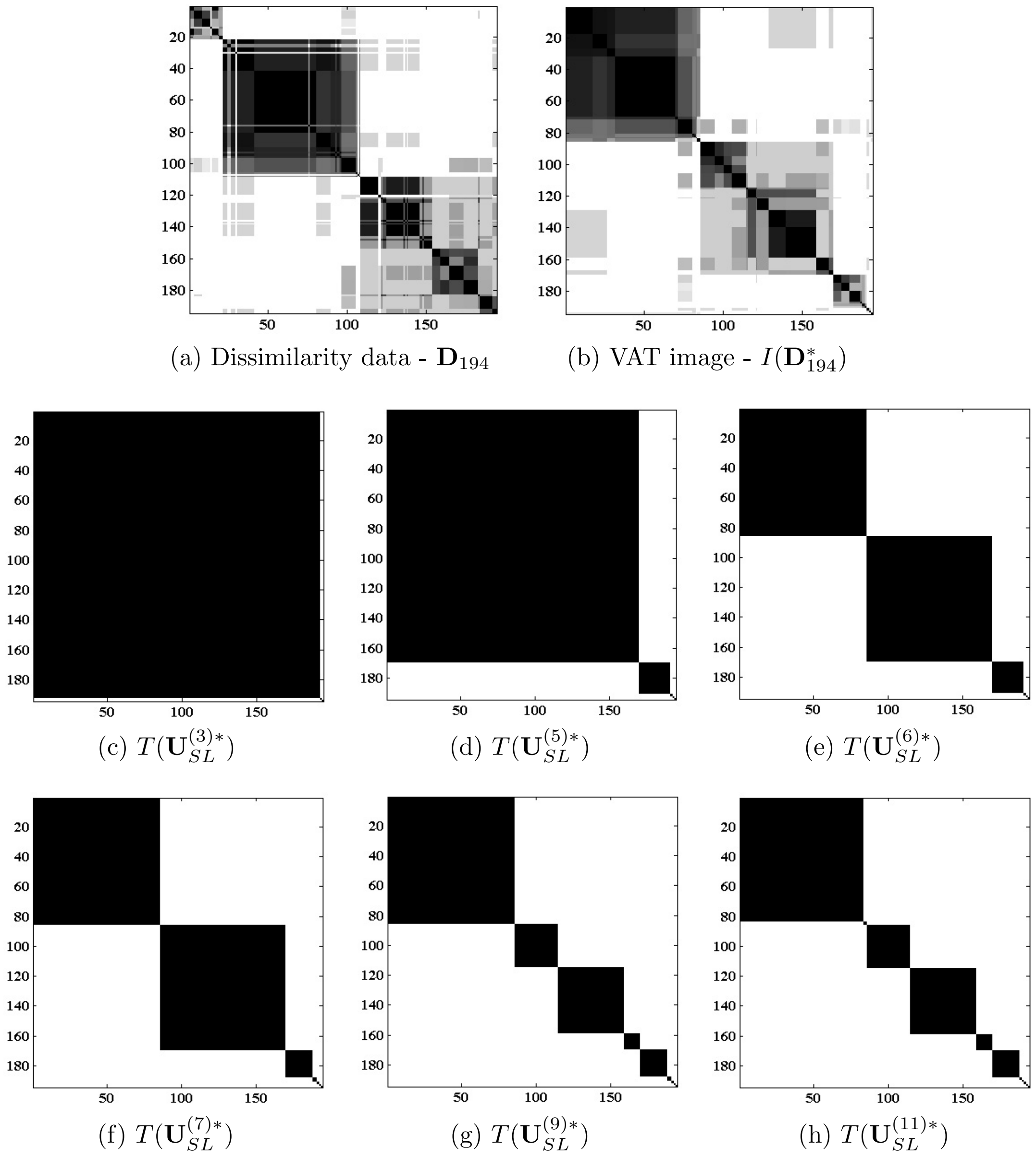


(h) $c = 6$ subtrees



(i) $T(\mathbf{U}_{SL}^{(6)*})$

Figure 3.2: Three parallel lines example: Images of the partition transformations $T(\mathbf{U}_{SL}^*)$ show that the SL partitions are aligned in the VAT ordered data.



(a) Dissimilarity data - $\mathbf{D}_{194}$ (b) VAT image - $I(\mathbf{D}^*_{194})$

(c) $T(\mathbf{U}^{(3)*}_{SL})$ (d) $T(\mathbf{U}^{(5)*}_{SL})$ (e) $T(\mathbf{U}^{(6)*}_{SL})$

(f) $T(\mathbf{U}^{(7)*}_{SL})$ (g) $T(\mathbf{U}^{(9)*}_{SL})$ (h) $T(\mathbf{U}^{(11)*}_{SL})$

Figure 3.3: Bioinformatics example: Images of the partition transformations $T(\mathbf{U}^*_{SL})$ show that the SL partitions are aligned in the VAT ordered data.

The images of the SL partition transformation $T(\mathbf{U}_{SL}^{(c)*})$ are displayed in Figs. 3.3(d-h). First, you can see that each partition of the VAT-reordered data is aligned. Second, the images show an interesting aspect of the SL clustering of this data. SL is known to be very susceptible to outliers and this is evident in these views. The $c = 3$ partition, displayed in Fig. 3.3(c), shows that SL clustering partitions the data into one large cluster and two very small clusters (located at the bottom right of the image). This is due to four gene products, which are outliers in this data set. Previous research has shown that these four outlier gene products, indexed as 120, 121, 30, and 107, actually cluster into three clusters, namely $\{120, 121\}$, $\{30\}$, and $\{107\}$ [Popescu et al., 2004]. The $c = 6$ partition, shown in Fig. 3.3(e), demonstrates that SL clustering partitions this data set into the expected three ENSEMBL families (the receptor precursor family is the first block, the collagen alpha chain is the second, and the myotubularins are the third, from the top of the image), plus the three outlier clusters. Figs. 3.3(f-h) illustrate that the families have further substructure. The tendency of the collagens to break up into three groups, as shown in Fig. 3.3(g), is supported by [Myllyharju and Kivirikko, 2004].

In summary, these three examples confirm the relationship of VAT and SL that is described analytically in Section 3.2.1—namely, that SL clusters of VAT-reordered data are aligned partitions.

3.3 VAT and Dunn's Validity Index

In the previous section it was proved that SL c-partitions will always be aligned partitions of the VAT reordered objects; hence, the image transformation $T(\mathbf{U}^*)$ will always show c dark blocks along the diagonal. The main function of VAT is to show cluster tendency as c dark blocks along the diagonal of the image of $\mathbf{D}^*$. However, VAT fails this task for the example in Figure 3.4. I address this problem with an analysis that shows that Dunn's index is a contrast measure of VAT images.

Dunn's index is a cluster validity measure that provides a metric as to how well the clusters that a clustering algorithm returns represent compact well-separated (CWS) clusters [Dunn, 1974]. For the set of all crisp c-partitions, $\mathbf{M}_{hcn}$, Dunn's index quantifies the CWS property of clusters defined by a partition $\mathbf{U}$ by computing

$$\alpha(c, \mathbf{U}) = \frac{\min_{1<q<c} \min_{1<r<c, r \neq q} \operatorname{dist}(\mathbf{C}_q, \mathbf{C}_r)}{\max_{1<p<c} \operatorname{diam}(\mathbf{C}_p)}, \tag{3.12}$$

where $\mathbf{C}_i$ is the i-th cluster as defined by the partition $\mathbf{U}$, $\operatorname{dist}(\mathbf{C}_q, \mathbf{C}_r)$ is the distance between two clusters, and $\operatorname{diam}(\mathbf{C}_p)$ is the cluster diameter. The diameter and distance functions, in terms of the dissimilarity data, are defined as

$$\operatorname{dist}(\mathbf{C}_q, \mathbf{C}_r) = \min_{o_i \in \mathbf{C}_q, o_j \in \mathbf{C}_r} D_{ij}, \tag{3.13}$$

$$\operatorname{diam}(\mathbf{C}_p) = \max_{o_i \in \mathbf{C}_p, o_j \in \mathbf{C}_p} D_{ij}, \tag{3.14}$$

where D_{ij} is the ijth element of $\mathbf{D}$.

The optimal partition for a given number of clusters c can be found by computing

$$\bar{\alpha}(c) = \max_{\forall \mathbf{U} \in \mathbf{M}_{hcn}} \alpha(c, \mathbf{U}). \tag{3.15}$$

For SL hierarchical clustering, and other deterministic clustering algorithms, there is only one partition for each value of c; hence, one does not need to compute (3.15). However, the relative validity of partitions at different values of c can be compared by comparing the respective values of (3.12) for each c-partition.

Dunn showed that clusters that have an $\alpha(c, \mathbf{U}) > 1$ are CWS clusters. He also showed that object data can be partitioned into c CWS clusters if and only if $\bar{\alpha}(c) > 1$—put more simply, clusters should be spaced far apart from each other relative to their size.

Proposition 3.3.1. *Let* $\mathbf{U} \in \mathbf{M}_{hcn}$ *such that* $\alpha(c, \mathbf{U}) > 1$*. Then the elements* D_{ij} *that correspond to* $T(\mathbf{U})_{ij} = 1$ *(between-cluster dissimilarity) are all strictly greater than the elements* D_{ij} *that correspond to* $T(\mathbf{U})_{ij} = 0$ *(within-cluster dissimilarity).*

Proof. Recall $T(\mathbf{U}) = [1] - \mathbf{U}^T\mathbf{U}$. The numerator of Dunn's index Eq.(3.12) can be rewritten by combining this transformation—described in Section 3.1—with Eq.(3.13),

$$\min_{1 \le q \le c} \min_{1 \le r \le c, r \ne q} \operatorname{dist}(\mathbf{C}_q, \mathbf{C}_r) = \min_{(T(\mathbf{U}))_{ij}=1} D_{ij}. \tag{3.16}$$

Similarly, the denominator of Eq.(3.12) can be rewritten,

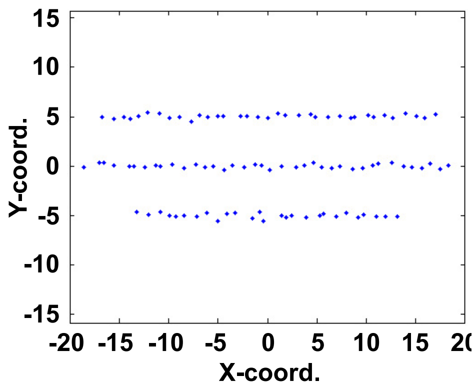
$$\max_{1 \le p \le c} \operatorname{diam}(\mathbf{C}_p) = \max_{(T(\mathbf{U}))_{ij}=0} D_{ij}. \tag{3.17}$$

Dunn's index can now be formulated in terms of $\mathbf{D}$ and $T(\mathbf{U})$,

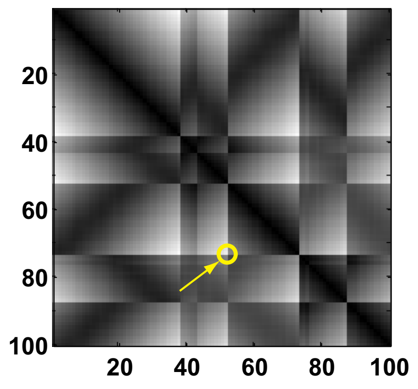
$$\alpha(c, \mathbf{U}) = \frac{\min_{(T(\mathbf{U}))_{ij}=1} D_{ij}}{\max_{(T(\mathbf{U}))_{ij}=0} D_{ij}}. \tag{3.18}$$

Equations (3.16) and (3.17) are always greater than or equal to 0; thus, if $\alpha(c, \mathbf{U}) > 1$ then the value of Eq.(3.16) is strictly greater than the value of Eq.(3.17), proving the proposition. □

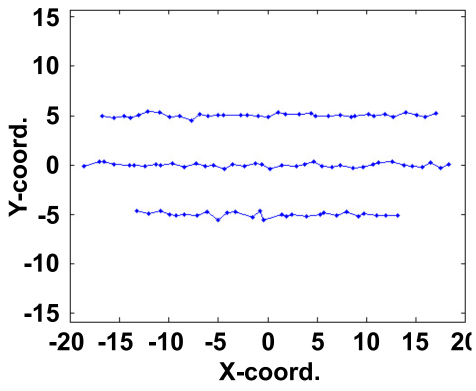
Remark 3.3.1. SL partitions $\mathbf{U}^*_{SL}$ are *always* aligned partitions of the VAT reordered objects. Hence, the image of $T(\mathbf{U}^*_{SL})$ appears as c dark blocks along the diagonal. Proposition 3.3.1 proves that the dissimilarity data between these blocks (cluster-cluster distances) will be strictly greater than the dissimilarity data within these blocks (in-cluster distances) if $\alpha(c, \mathbf{U}^*_{SL}) > 1$. Evidently, the VAT image will have



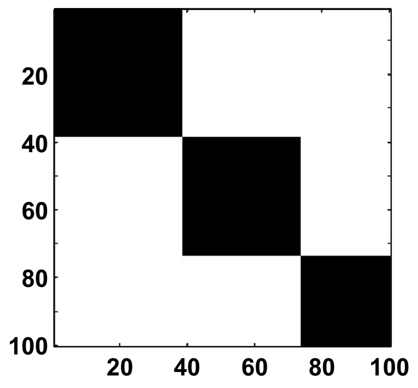
(a) 100 numerical objects - $\mathbf{X}$



(b) VAT image - $I(\mathbf{D}^*)$



(c) $(c = 3)$ SL partition $\mathbf{U}^*_{SL}$ shown as connected subtrees



(d) Image of transformation $T(\mathbf{U}^*_{SL})$

Figure 3.4: Three parallel lines example shows that VAT can fail to show the visually-preferable cluster tendency of 3 clusters. However, the image of the partition transformation $T(\mathbf{U}^*_{SL})$ shows that the SL 3-partition is aligned in the VAT ordered data.

c dark blocks if $\alpha(c, \mathbf{U}^*_{SL}) > 1$. Furthermore, the value of Dunn's index provides a measure of the "blockiness" of the VAT image—the greater the value of Dunn's index, the greater the contrast of the blocks in the VAT image.

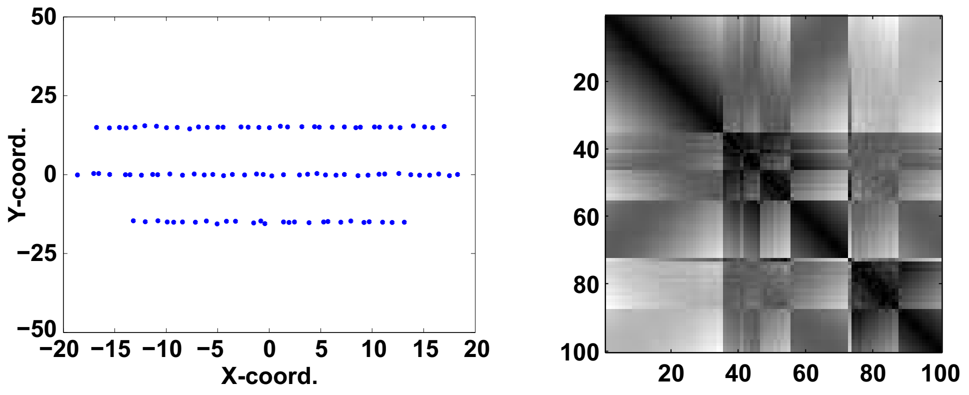
The three dark blocks in Fig. 3.4(d) represent the grouping of the objects according to the 3-partition shown in Fig. 3.4(c). The three dark blocks show the elements of $\mathbf{D}^*$ within the clusters and the white regions show the elements of $\mathbf{D}^*$ between the clusters. Hence, the denominator of Eq.(3.12) is the maximum element in $\mathbf{D}^*$ within the three dark blocks (the diameter of the middle line), which is shown on Fig. 3.4(b).

Similarly, the numerator of Eq.(3.12) is the minimum element in $\mathbf{D}^*$ in the white regions between the three dark blocks. The value of Dunn's index for the 3-partition shown in Fig. 3.4(c) is 0.12 and VAT fails to show ($c = 3$).

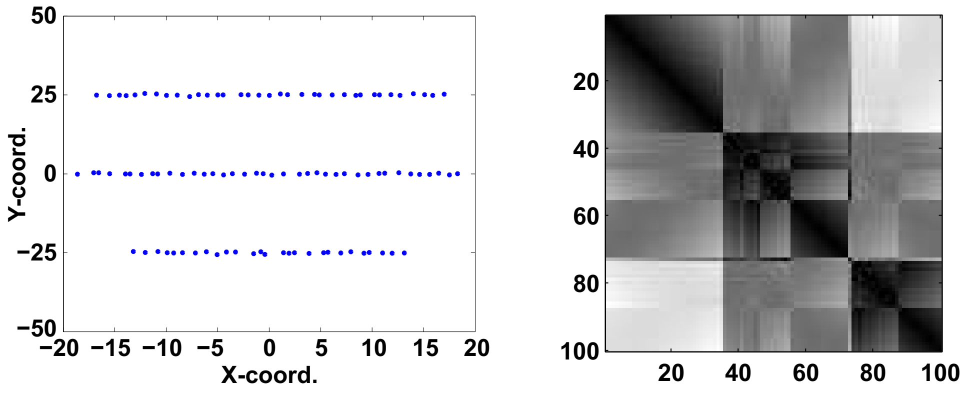
Figure 3.5 shows the VAT image for the same type of example shown in Fig. 3.2(a); each view shows the three lines moved progressively farther apart. Figures 3.5(a,b) shows cases where the SL 3-partition produces a Dunn's index < 1 —which is *still* the maximum possible value over all $\mathbf{U} \in \mathbf{M}_{hcn}$. Figure 3.5(c) illustrates the case where Dunn's index > 1; hence, the elements of $\mathbf{D}^*$ within the three dark blocks are strictly greater than the background elements of $\mathbf{D}^*$. I argue that the VAT image in Fig. 3.5(c) is the only image that undoubtedly suggests three clusters. This example gives empirical evidence to support Proposition 3.3.1 and Remark 3.3.1.

My analysis and the example shown in Fig. 3.5 demonstrate that the effectiveness of VAT in showing cluster tendency is tied directly to Dunn's index. Dunn's index provides a measure of contrast between the blocks on the VAT image diagonal and the background regions. Although the examples shown are for one artificial data set, Proposition 3.3.1 proves that this result is true for all data sets. Proposition 3.3.1 also provides a fast and elegant way to implement Dunn's index in applications, such as MATLAB, or on a *Graphics Processing Unit* (GPU). Additionally, this analysis poses the question of the possibility to adapt Dunn's index to the empty-cluster or "noise" cluster case.

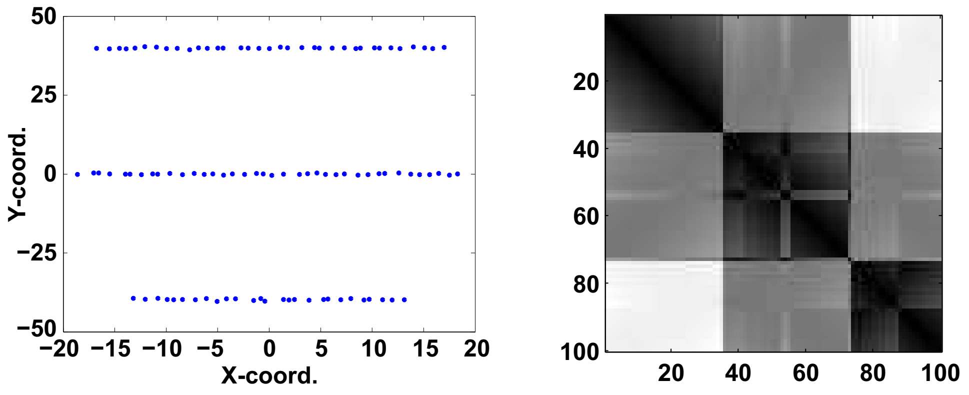
There are many questions that are not answered by this analysis: (i) What if more than one $\mathbf{U} \in \mathbf{M}_{hcn}$ produces a Dunn's index > 1? In such cases, of which partition $\mathbf{U}$ will VAT show the cluster tendency? (ii) The VAT image is meant to be interpreted by a human; hence, how does human perception of the "blockiness" relate to this analysis? (iii) How does this analysis apply to VAT-based algorithms such as *Visual Cluster Validity* [Hathaway and Bezdek, 2003, Huband and Bezdek, 2008]?



(a) Dunn's Index = 0.39



(b) Dunn's Index = 0.66



(c) Dunn's Index = 1.06

Figure 3.5: Three examples of object data (left) and corresponding VAT images (right) to show that Dunn's Index is a "blockiness" measure of VAT images. At which value of Dunn's index do you see a cluster tendency of 3 in the VAT image?

3.4 VAT = iVAT + Distance Transform

There are many cases where VAT fails to show the cluster tendency of data sets that have a visually preferable number of clusters, e.g., the Three Lines data set shown in Figs. 3.4(a,b). To combat this issue, Wang et al. [2010] proposed an *improved*-VAT (iVAT) algorithm that uses a path-based distance measure developed in Fisher et al. [2001]. By using the path-based distance, iVAT is often able to improve the visual contrast of the dark blocks along the VAT diagonal. However, iVAT, as originally proposed in [Wang et al., 2010], is computationally expensive, $O(n^3)$—as opposed to $O(n^2)$ of VAT.

Here I will show that iVAT is essentially a distance transform of the original VAT image. I denote $\mathbf{D}'$ as $\mathbf{D}'^*$ to indicate the VAT-ordered path-based distance matrix. Algorithm 3.4.1 outlines my recursive formulation of iVAT from the VAT-reordered dissimilarity matrix $\mathbf{D}^*$. Line 1 requires $(r-1)$ comparisons and Line 3 requires $(r-2)$ comparisons, for a total of $(2r-3)$ operations. The total number of operations in Algorithm 3.4.1 is thus $(2n^2-3n)$, which is $O(n^2)$ complexity. I was able to reduce the complexity by using VAT itself as a preprocessing step. In contrast to the iVAT formulation in [Wang et al., 2010], I start with VAT and then transform the VAT-reordered dissimilarity data into the iVAT image with my recursive algorithm.

Algorithm 3.4.1: Recursive calculation of iVAT image

```
  Input: $\mathbf{D}^*$ - VAT-reordered dissimilarity matrix
  Data: $\mathbf{D}'^* = [0]^{n\times n}$
  for $r = 2,\ldots,n$ do
1     $j = \arg\min_{k=1,\ldots,r-1} D^*_{rk}$
2     $D'^*_{rc} = D^*_{rc}, c = j$
3     $D'^*_{rc} = \max\left\{D^*_{rj}, D'^*_{jc}\right\}, c = 1,\ldots,r-1, c \neq j$
  $\mathbf{D}'^*$ is symmetric, thus $D'^*_{rc} = D'^*_{cr}$.
```

Without loss of generality, let the objects in any VAT-reordered object data $\mathbf{O}^*$

also represent the vertices of a fully connected graph, where the individual vertices are denoted as o_i^*, $i = 1, \ldots, n$. For ease of notation, I denote the vertices simply by the objects' indexes, $i = 1, \ldots, n$. The edge weights of this graph are computed by your chosen vector norm or dissimilarity function d,

$$\text{edge weight}_{ij} = d(i, j) = D_{ij}^*. \tag{3.19}$$

Then,

$$d_{min}(I, J) = \min_{\forall i \in I, \forall j \in J} D_{ij}^* \tag{3.20}$$

Lemma 3.4.1. *Consider a vertex $k, 1 < k < n$, and the sets of vertices, $I = \{1, \ldots, k-1\}$ and $J = \{k+1, \ldots, n\}$. Then,*

$$d_{min}(I, k) \leq d_{min}(I, J). \tag{3.21}$$

Proof. Recall that VAT is a special case of Prim's algorithm and, thus, computes the *minimum-spanning-tree* (MST) of a set of vertices (objects) by adding the vertex which is closest to the already ordered vertices. By the definition of VAT, (3.21) is true. □

Remark 3.4.1. In the case where each edge has a unique weight, then $d_{min}(I, k) < d_{min}(I, J)$ can be shown to be true (see Section 3.2).

The next lemma proves that line 2 in Algorithm 3.4.1 is valid. Note that $\mathbf{D}^*$ and $\mathbf{D}'$ are symmetric distance matrices.

Lemma 3.4.2. *Consider the vertices $k, 1 < k \leq n$, and $l, 1 \leq l < k$, and the sets of vertices $I = \{1, \ldots, k-1\}$ and $J = \{k+1, \ldots, n\}$, where $J = \emptyset$ if $k = n$. If*

$$D_{kl}^* = d_{min}(I, k), \tag{3.22}$$

then the path-based distance (2.4) between k and l is

$$D'_{kl} = D^*_{kl}. \tag{3.23}$$

Proof. Lemma 3.4.1 shows that $d_{min}(I,k) \leq d_{min}(I,J)$, which can be extended to $D^*_{kl} = d_{min}(I,k) \leq d_{min}(I,J)$. Thus,

$$D^*_{kl} \leq \min_{p} \max_{1 \leq h < |p|} D_{p[h]p[h+1]}, \tag{3.24}$$

for all paths $p \in P_{kl}$ that include a vertex in J. Equation (3.22) shows that

$$D^*_{kl} \leq \min_{p} \max_{1 \leq h < |p|} D_{p[h]p[h+1]}, \tag{3.25}$$

for all paths $p \in P_{kl}$ that include a vertex in I. Thus, $D'_{kl} = D^*_{kl}$. If $k = n$, consider (3.25), and it is easy to see that the lemma holds true. □

Remark 3.4.2. Notice that D^*_{kl} in Lemma 3.4.2 is the weight of the MST edge that connects vertex k to the sub-tree I.

The next lemma proves that line 3 of the recursive algorithm is valid.

Lemma 3.4.3. *Consider the vertices $k, 1 < k \leq N$, and $l, 1 \leq l < k$, and the sets of vertices $I = \{1, \ldots, k-1\}$ and $J = \{k+1, \ldots, n\}$, where $J = \emptyset$ if $k = n$. If*

$$D^*_{kl} > d_{min}(I,k) \tag{3.26}$$

and

$$s = \arg\min_{1 \leq t < k} D^*_{kt}, \tag{3.27}$$

then the path-based distance (2.4) between k and l,

$$D'_{kl} = \max\{D^*_{ks}, D'_{sl}\}. \tag{3.28}$$

Proof. Lemma 3.4.3 is proved by showing that the path $p \in P_{kl}$ that produces the minimum $\left\{\max_{1\leq h<|p|} D^*_{p[h]p[h+1]}\right\}$ is the path along the MST edges between vertices k and l. Lemma 3.4.2 shows that $D^*_{ks} \leq d_{min}(I, J)$. Thus, all paths $p \in P_{kI}$ through the vertices J have an edge with a weight $\geq D^*_{ks}$. In other words, D^*_{ks} is the least costly path from vertex k to the sub-tree I, where $l \in I$. By definition, D'_{sl} is the value of the maximum edge weight along the least costly path from s to l, thus $D'_{kl} = \max\{D^*_{ks}, D'_{sl}\}$. □

Remark 3.4.3. Notice that D^*_{ks} is the weight of the $(k-1)$th edge added the MST by Prim's algorithm. Additionally, it can be shown that D'_{sl} is the weight of the $(l-1)$th edge added to the MST. Thus, all the distance values in $\mathbf{D'}$ are the weights of the maximum MST edge that is traversed between vertices k and l. This logic can also be extended to show that the path p which produces each value of $\mathbf{D'}$ is the path along the MST.

Equations (3.25) and (3.28) are applied recursively, starting with $k = 2$, to calculate $\mathbf{D'}$ from $\mathbf{D}^*$. Now I show that the result of this recursive calculation is an iVAT image, $\mathbf{D'} = \mathbf{D'}*$.

The properties of a VAT image are:

1. The first vertex in the ordering is one of the set of vertices that satisfy

$$k = \arg\max_{\forall i} \left\{\max_{\forall j} D_{ij}\right\}. \tag{3.29}$$

2. The ordering of the vertices is that which could be computed using Prim's

algorithm.

First, I show that $\mathbf{D}'$ satisfies the first VAT property by showing that the first row in $\mathbf{D}'$—the distances between vertex 1 and all other vertices—contains an element that is the maximum of $\mathbf{D}'$.

Lemma 3.4.4. $\mathbf{D}'$ *satisfies*

$$\max_{\forall i} D'_{1i} = \max_{\forall i,j} D'_{ij}. \tag{3.30}$$

Proof. The path-based distance D'_{1i} is the value of the maximum weighted edge along the MST path between vertices 1 and i. Hence, $\max_{\forall i} D'_{1i}$ is the value of the maximum weighted MST edge because all MST edges are traversed on at least one path $p \in P^{MST}_{1,\forall j}$, where $P^{MST}_{1,\forall j}$ denotes all possible paths between vertex 1 and all other vertices that are on the MST. Additionally, all the elements of $\mathbf{D}'$ are MST edge weights. Thus, $\max_{\forall i} D'_{1i} = \max_{\forall i,j} D'_{ij}$. □

Remark 3.4.4. An interesting consequence of Lemma 3.4.4 is that it can be simply modified to show that *any* vertex can be chosen as the initial vertex in iVAT.

Next, I show that $\mathbf{D}'$ satisfies the second VAT property by first defining

$$d'_{min}(I,J) = \min_{\forall i\in I, \forall j \in J} D'_{ij}. \tag{3.31}$$

Lemma 3.4.5. *Consider the vertex* $1 < k < n$ *and the sets of vertices,* $I = \{1, \ldots, k-1\}$ *and* $J = \{k+1, \ldots, n\}$*. The dissimilarity matrix* $\mathbf{D}'$ *satisfies*

$$d'_{min}(I,k) \le d'_{min}(I,J). \tag{3.32}$$

Notice that this Lemma essentially shows that the property of $\mathbf{D}^*$ proven in Lemma 3.4.1 applies to $\mathbf{D}'$.

Proof. Consider the MST edges that must be cut in order to produce the MST subtree I and the MST subtrees in J. By definition of VAT and Prim's algorithm the weight of the MST edge that connects k to I is less than or equal to the weights of the MST edges that connect I to the subtrees in J (if this was not true, then the vertices would be differently ordered by VAT). All MST paths from I to J must pass through the MST edges that are cut, thus the lower-bound on $d'_{min}(I, J)$ is the weight of these edges. The lower-bound on $d'_{(min)}(I, k)$ is the weight of the MST edge that connects k to I; hence, $d'_m in(I, k) \leq d'_{min}(I, J)$.

Note that a special case to consider is where there is one MST subtree in J and this is connected to I through an MST edge to vertex k. In this special case, it is easy to see that all MST paths from I to J must pass through k and thus (3.32) is true. □

Remark 3.4.5. There are many possible VAT-reorderings of $\mathbf{D}'$ because there are many *ties* in the path-based distances and any object could be chosen as the initial object. Lemmas 3.4.4 and 3.4.5 show that $\mathbf{D}'$, as calculated by the recursive formulas in Lemmas 3.4.2 and 3.4.3, is already in one possible VAT reordering. And, arguably, this ordering of $\mathbf{D}'$ is the "best" reordering because it is also the VAT-reordering of the original dissimilarity matrix.

3.4.1 Discussion

The iVAT algorithm proposed in [Wang et al., 2010] was shown to improve the quality of VAT images, which ideally improves the interpretability of the clustering tendency. The recursive formulation I propose significantly reduces the computational complexity of the iVAT algorithm; my formulation has a complexity of $O(n^2)$, compared to $O(n^3)$ for the formulation presented in [Wang et al., 2010]. Moreover, my algorithm produces both the VAT and iVAT images, which is not the case for the original iVAT method.

In every test of VAT versus iVAT I have either seen or run, the iVAT image was at least as good (visually) as the VAT image. My conjecture is that iVAT images will always be equal to or superior to VAT images, but to date I have not discovered a way to prove this. Because this is a subjective evaluation, it may not be provable at all. Nonetheless, until a counterexample is discovered, I believe that with the recursive formulation, there is no reason not to default to iVAT in every instance. Section A.1 in the Appendix shows more examples of iVAT images, specifically of noisy large-scale data sets.

Additionally, I have inserted the recursive formulation of iVAT Eq.(2.4) into the *improved* co-VAT (co-iVAT) algorithm for visual assessment of clustering tendency in rectangular dissimilarity data, which is proposed in Section 4.2.2.

Future directions include extending the iVAT algorithm to the scalable versions of VAT and co-VAT, *scalable VAT* (sVAT) [Hathaway et al., 2006], bigVAT [Huband et al., 2005], and *scalable co-VAT* (scoVAT) [Park et al., 2009]. These algorithms work with very-large data, or data that is unloadable on standard computers.

This chapter began by presenting a relationship between VAT and SL clustering. Essentially, this relationship suggests that VAT images can be used to partition data into SL-like clusters; this is the problem I address in the next chapter of this dissertation.

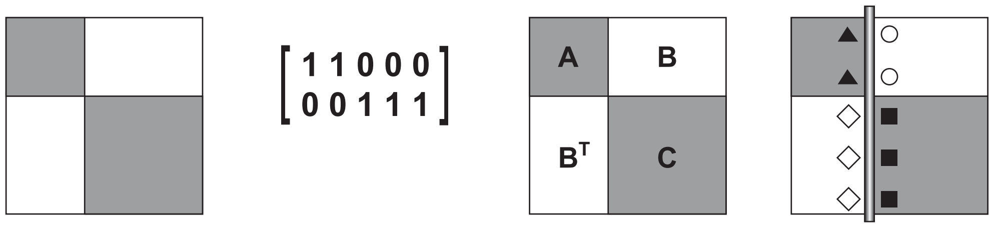
Chapter 4

Relational Clustering Algorithms

4.1 Clustering in Ordered Dissimilarity Data

Assume as input a normalized (entries between 0 and 1) dissimilarity matrix $\mathbf{D}^*$ (equivalently, $I(\mathbf{D}^*)$) that is symmetric with diagonal elements that are zero. The superscript (*) indicates that $\mathbf{D}$ has been reordered by some algorithm to produce a "VAT-like" image, as in Fig. 2.4. The important property of $I(\mathbf{D}^*)$ is that it has, beginning in the upper left corner, dark blocks along its diagonal. Accordingly, the search through $\mathbf{M}_{hcn}$ for each c under consideration is constrained to those partitions that mimic the blocky structure in $I(\mathbf{D}^*)$, namely the aligned partitions $\mathbf{M}^*_{hcn}$ described in Section 3.1.

The important characteristics of $I(\mathbf{D}^*)$ that I shall exploit for finding a $\mathbf{U}$ that seems to match it are: (i) the contrast between the dark blocks along the main diagonal and the lighter off-diagonal blocks; and (ii) the visually apparent edges of those dark blocks. The algorithm generates candidate partitions in $\mathbf{M}^*_{hcn}$ and tests their fit to the clusters suggested by the aligned dark blocks in $I(\mathbf{D}^*)$. To accomplish this, I define an objective function on $\mathbf{M}^*_{hcn}$ that computes a measure of two properties



(a) Ideal $I(\mathbf{D}^*)$ (b) Optimal partition $\mathbf{U} \in \mathbf{M}^*_{h25}$ (c) $E_{\text{sq}}(\mathbf{U})$ "squareness" (d) $E_{\text{edge}}(\mathbf{U})$ "edginess"

Figure 4.1: The components of the CLODD objective function $E(\mathbf{U})$: $E_{\text{sq}}(\mathbf{U})$ measures the contrast of the dark blocks along diagonal compared to the off-diagonal light blocks; $E_{\text{edge}}(\mathbf{U})$ measures the edge contrast of the dark blocks.

of blocky images $I(\mathbf{D}^*)$ — "squareness" and "edginess". Figure 4.1(a) shows an idealized case of $I(\mathbf{D}^*)$ for $c = 2$ which, for illustration purposes, assumes that $n = 5$.

Figure 4.1(b) shows the presumably optimal aligned partition that provides the best fit to the image in 4.1(a). Figure 4.1(c) shows the "squareness" component of the objective function that measures the contrast between diagonal dark blocks $\mathbf{A}$ and $\mathbf{C}$ and the off-diagonal blocks $\mathbf{B}$ and $\mathbf{B^T}$ according to the $\mathbf{U}$ in 4.1(b). An intuitively appealing measure is the difference of the average dissimilarity values *between* apparent clusters (i.e. dissimilarities in [($\mathbf{A}$,$\mathbf{B}$)] and [($\mathbf{B^T}$,$\mathbf{C}$)]) and those *within* apparent clusters (i.e. dissimilarities in [($\mathbf{A}$,$\mathbf{A}$)] and [($\mathbf{C}$,$\mathbf{C}$)]). Let $\mathbf{U}$ be a candidate partition in $\mathbf{M}^*_{hcn}$, let $\{\mathbf{O}_i : 1 \leq i \leq c\}$ be the crisp c-partition of $\mathbf{O}$ corresponding to $\mathbf{U}$. The cardinality $|\mathbf{O}_i| = n_i \forall i$, and the membership $o_s \in \mathbf{O}_i$ is abbreviated simply as $s \in i$. With these heuristics, the "squareness" component of the objective function for a given $\mathbf{D}^*$ is:

$$E_{\text{sq}}(\mathbf{U};\mathbf{D}^*) = \underbrace{\left(\frac{\displaystyle\sum_{i=1}^{c} \sum_{s \in i, t \notin i} d^*_{st}}{\displaystyle\sum_{i=1}^{c}(n-n_i)n_i} \right)}_{\substack{\text{ave. dissimilarity } \textit{between} \text{ dark} \\ \text{and non-dark regions in } I(\mathbf{D}^*)}} - \underbrace{\left(\frac{\displaystyle\sum_{i=1}^{c} \sum_{s,t \in i, s \neq t} d^*_{st}}{\displaystyle\sum_{i=1}^{c}(n_i^2-n_i)} \right)}_{\substack{\text{ave. dissimilarity } \textit{within} \\ \text{dark regions in } I(\mathbf{D}^*)}} . \quad (4.1)$$

Good candidate partitions $\mathbf{U}$ should maximize Eq.(4.1). This equation is a measure of contrast between the on-diagonal dark blocks and the off-diagonal non-dark blocks.

The "edginess" of the dark blocks in $\mathbf{D}^*$ is computed by averaging the values of the first order estimate of the horizontal digital gradient across each vertical boundary imposed by a candidate $\mathbf{U}$ in $\mathbf{M}^*_{hcn}$. Figure 4.1(d) shows the edges that are considered for this part of the objective function. The symbols along the vertical boundary separating the dark from the non-dark blocks represent dissimilarity values in the columns of $\mathbf{D}^*$ adjacent to the boundary. The "edginess" value for the example in 4.1(d) is computed by

$$E_{\text{edge}}(\mathbf{U}) = \left(\frac{\sum |\bigcirc - \triangle| + \sum |\Diamond - \Box|}{2+3} \right).$$

For the c blocks in $\mathbf{D}^*$, there are $(c-1)$ interior vertical boundaries between dark blocks and adjacent blocks of lighter intensities. Each vertical edge spans the right face of an upper block and the left face of the block immediately below it. Let $\mathbf{U} = \{n_1 : \ldots : n_c\} \in \mathbf{M}^*_{hcn}$, a candidate aligned partition. For $j = 1$ to $c-1$, let

$m_j = \sum_{k=1}^{j} n_k$, and $m_0 = 1$. The "edginess" measure is defined as

$$E_{\text{edge}}(\mathbf{U}; \mathbf{D}^*) = \frac{1}{c-1} \sum_{j=1}^{c-1} \frac{\sum_{i=m_{j-1}}^{m_j} |d^*_{i,m_j} - d^*_{i,m_j+1}| + \sum_{i=m_j+1}^{m_{j+1}} |d^*_{i,m_j} - d^*_{i,m_j+1}|}{n_j + n_{j+1}}. \tag{4.2}$$

Good candidate partitions **U** should maximize Eq.(4.2). Although this equation looks complicated, it is merely the average horizontal gradient across vertical edges separating dark blocks from non-dark blocks in $I(\mathbf{D}^*)$. Good candidate partitions **U** maximize both Eqs. (4.1) and (4.2), which allows us to add them together to produce a composite objective function. To make the resulting sum flexible in terms of the balance between contrast and edginess, I use the convex combination of Eqs. (4.1) and (4.2). Let α be the *mixing coefficient*, and

$$E_\alpha(\mathbf{U}; \mathbf{D}^*) = \alpha E_{\text{sq}}(\mathbf{U}; \mathbf{D}^*) + (1-\alpha) E_{\text{edge}}(\mathbf{U}; \mathbf{D}^*); 0 \leq \alpha \leq 1. \tag{4.3}$$

If contextual information is unavailable to suggest that one factor, contrast or edginess, is more important than the other, one may take $\alpha = 1/2$, which gives equal weight to contrast and edginess in $\mathbf{D}^*$.

The final component of the objective function controls the size of the smallest cluster allowed in the search over $\mathbf{M}^*_{hcn}$. I use the spline function,

$$s(x, a) = \begin{cases} 0 & ; x \leq 1 \\ 2\left(\frac{x}{a}\right)^2 & ; 1 < x \leq \frac{a}{2} \\ 1 - 2\left(\frac{a-x}{a}\right)^2 & ; \frac{a}{2} < x < a \\ 1 & ; a \leq x \end{cases}, \tag{4.4}$$

for this purpose. This function is a typical s-curve valued in $[0, 1]$ with points of

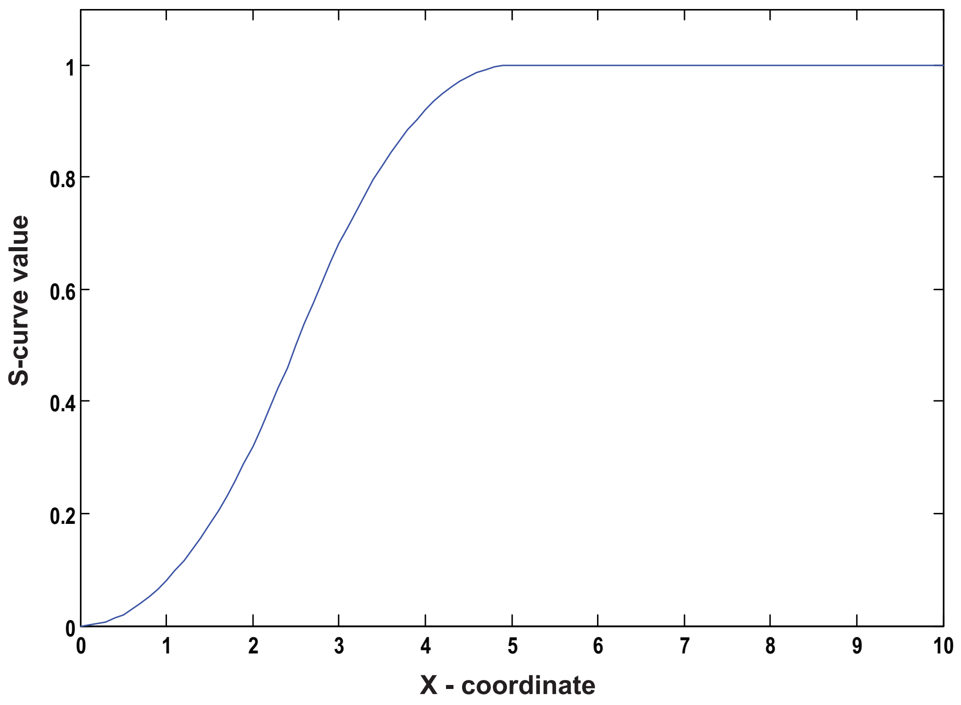


Figure 4.2: S-curve, Eq.(4.4), with $a = 5$

inflection at $a/2$ and a. For $\mathbf{U} = \{n_1 : \ldots : n_c\} \in \mathbf{M}^*_{hcn}$, I set the inflection points by choosing $a = \gamma n$, $2/n < \gamma < 1$, and then evaluate s at $x = \min_{1 \le i \le c}\{n_i\}$.

Finally, the function in Eq.(4.3) is multplied by

$$S_\gamma(\mathbf{U}) = s\left(\min_{1 \le i \le c}\{n_i\}, \gamma n\right). \tag{4.5}$$

This scales Eq.(4.3) in a way that enables us to damp very small clusters in candidates partitions when none are apparent in $\mathbf{D}^*$. If one of the candidate partitions contains a cluster that has fewer than $a = \gamma n$ objects, then CLODD penalizes the value of the objective function of this candidate partition. Essentially, γ sets the the expected smallest cluster size for a particular data set.

The objective function is now complete, so an optimal partition of $\mathbf{D}^*$ is defined as one that maximizes

$$E(\mathbf{U};\mathbf{D}^*) = s\left(\min_{1\le i\le c}\{n_i\}, \gamma n\right) \cdot E_\alpha(\mathbf{U};\mathbf{D}^*) = S_\gamma(\mathbf{U}) \cdot E_\alpha(\mathbf{U};\mathbf{D}^*). \qquad (4.6)$$

Last, I want to search for the best partition at various values of c, so let $\mathbf{C} = \{2, 3, \ldots, c_{max}\}$. The optimization problem that the CLODD algorithm attempts to solve is

$$\max_{\mathbf{U}\in\mathbf{M}^*_{hcn}, c\in\mathbf{C}}\{E(\mathbf{U};\mathbf{D}^*)\} \qquad (4.7)$$

An approximate global solution of Eq.(4.7) is denoted by $\mathbf{U}_{c^*}$. One needs to choose two model parameters (α, γ), and then solve the optimization problem in Eq.(4.7). Before I turn to the solution of Eq.(4.7), I give an example that illustrates the basic ideas of this approach.

Example 4.1.1. Shown in Fig. 4.3(a,c) is a matrix $\mathbf{D}$ and the image $I(\mathbf{D})$ of dissimilarities between five objects $\mathbf{O} = \{o_1, \ldots, o_5\}$. Figure 4.3(b,d) shows the VAT reordering $\mathbf{D}^*$ of $\mathbf{D}$, and the VAT image $I(\mathbf{D}^*)$ corresponding to this reordering.

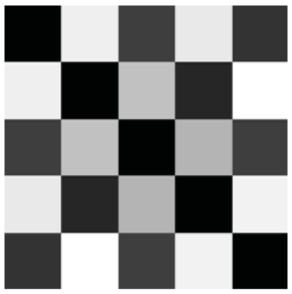
Visual inspection of $I(\mathbf{D})$ does not reveal whether the objects represented by pair-wise dissimilarities in $\mathbf{D}$ might form clusters in $\mathbf{O}$. In addition, it is easy to see that cluster structure *is* suggested by the two dark block in the VAT image $I(\mathbf{D}^*)$. The strong impression given by $I(\mathbf{D}^*)$ is that this is an instance for which the ideal case is shown in Fig. 4.1(a). Thus, the aligned 2-partition of $\mathbf{O}$ that should provide a best match to $I(\mathbf{D}^*)$ is the one shown in Fig. 4.1(b) corresponding to $o = \{o^*_1, o^*_2\} \cup \{o^*_3, o^*_4, o^*_5\}$. At this point, VAT has done its job. One could apply CCE [Sledge et al., 2008] or autoVAT [Wang et al., 2009] to $I(\mathbf{D}^*)$, and those algorithms would return the value $c = 2$, telling us to look for two clusters in $\mathbf{O}$. Despite this, these algorithms (VAT, CCE, and/or autoVAT) still have not defined cluster partitions. To obtain the $\mathbf{U}$ in Fig. 4.1(b) that is suggested by $I(\mathbf{D}^*)$, I apply

$$\mathbf{D} = \begin{bmatrix} 0 & 0.73 & 0.19 & 0.71 & 0.16 \\ 0.73 & 0 & 0.59 & 0.12 & 0.78 \\ 0.19 & 0.59 & 0 & 0.55 & 0.19 \\ 0.71 & 0.12 & 0.55 & 0 & 0.74 \\ 0.16 & 0.78 & 0.19 & 0.74 & 0 \end{bmatrix}$$

(a) Dissimilarity matrix - $\mathbf{D}$

$$\mathbf{D} = \begin{bmatrix} 0 & 0.12 & 0.59 & 0.73 & 0.78 \\ 0.12 & 0 & 0.55 & 0.71 & 0.74 \\ 0.59 & 0.55 & 0 & 0.19 & 0.19 \\ 0.73 & 0.71 & 0.19 & 0 & 0.16 \\ 0.78 & 0.74 & 0.19 & 0.16 & 0 \end{bmatrix}$$

(b) VAT reordered dissimilarity matrix - $\mathbf{D}^*$


(c) Image - $I(\mathbf{D})$

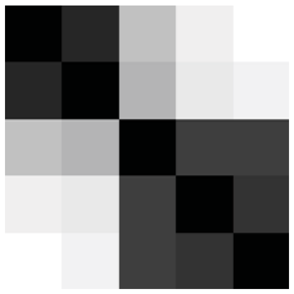

(d) VAT image - $I(\mathbf{D}^*)$

Figure 4.3: Dissimilarity data used in CLODD example 4.1.1. VAT image shows a cluster tendency of 2.

CLODD to $\mathbf{D}^*$.

To see how the CLODD objective function $E(\mathbf{U};\mathbf{D}^*)$ compares candidates, consider the aligned 2-partitions,

$$\mathbf{U} = \{2:3\} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

and

$$\mathbf{V} = \{3:2\} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

and their transformations under f and g,

$$f(\mathbf{U}) = \mathbf{U}^T\mathbf{U} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}; g(\mathbf{U}) = [1] - f(\mathbf{U}) = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad (4.8)$$

$$\mathbf{U} \leftrightarrow \{2:3\} \in \mathbf{M}^*_{h25} \qquad\qquad \mathbf{U} \leftrightarrow \{3:2\} \in \mathbf{M}^*_{h25}$$

$$\mathbf{D}^* = \left[\begin{array}{cc|ccc} 0 & 0.12 & 0.59 & 0.73 & 0.78 \\ 0.12 & 0 & 0.55 & 0.71 & 0.74 \\ \hline 0.59 & 0.55 & 0 & 0.19 & 0.19 \\ 0.73 & 0.71 & 0.19 & 0 & 0.16 \\ 0.78 & 0.74 & 0.19 & 0.16 & 0 \end{array}\right] \quad \Bigg| \quad \mathbf{D}^* = \left[\begin{array}{ccc|cc} 0 & 0.12 & 0.59 & 0.73 & 0.78 \\ 0.12 & 0 & 0.55 & 0.71 & 0.74 \\ 0.59 & 0.55 & 0 & 0.19 & 0.19 \\ \hline 0.73 & 0.71 & 0.19 & 0 & 0.16 \\ 0.78 & 0.74 & 0.19 & 0.16 & 0 \end{array}\right]$$

Figure 4.4: Boundaries imposed on $\mathbf{D}^*$ by choosing $\mathbf{U} = \{n_1 : n_2\} \in \mathbf{M}^*_{h25}$

$$f(\mathbf{V}) = \mathbf{V}^T\mathbf{V} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}; g(\mathbf{V}) = [1] - f(\mathbf{V}) = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}. \quad (4.9)$$

The blocks of 1's in $f(\mathbf{U}) = \mathbf{U}^T\mathbf{U}$ and $f(\mathbf{V}) = \mathbf{V}^T\mathbf{V}$ show the regions in $\mathbf{D}^*$ over which the CLODD calculations are made (as do $g(\mathbf{U})$ and $g(\mathbf{V})$). The partition parameters $\{2:3\}$ and $\{3:2\}$ set up "boundaries" in $\mathbf{D}^*$ as shown in Fig. 4.4.

For this example, equations (4.1) and (4.2) yield,

$$E_{\text{sq}}(\mathbf{U};\mathbf{D}^*) = (0.59+0.73+0.78+0.55+0.71+0.74)/6 - (0.12+0.19+0.19+0.16)/4 = 0.52,$$

$$E_{\text{sq}}(\mathbf{V};\mathbf{D}^*) = (0.73+0.78+0.71+0.74+0.19+0.19)/6 - (0.12+0.59+0.55+0.1)/4 = 0.20,$$

$$\begin{aligned} E_{\text{edge}}(\mathbf{U};\mathbf{D}^*) &= [|0.12-0.59| + |0-0.55| + \\ &\quad |0.55-0| + |0.71-0.19| + |0.74-0.19|]/5 \\ &= 0.53, \end{aligned}$$

$$\begin{aligned} E_{\text{edge}}(\mathbf{U};\mathbf{D}^*) &= [|0.59-0.73|+|0.55-0.71|+|0-0.19|+ \\ &\quad |0.19-0|+|0.19-0.16|]/5 \\ &= 0.14. \end{aligned}$$

In this example the smallest $n_i = 2$ and $n = 5$ for both $\mathbf{U}$ and $\mathbf{V}$, so the spline factor in Eq.(4.4) has the same value for any choice of γ; without loss, take $S_\gamma(\mathbf{U}) = 1$. Choosing $\alpha = 0.5$ in Eq.(4.6 I arrive at the final values,

$$E(\mathbf{U};\mathbf{D}^*) = E_{0.5}(\mathbf{U};\mathbf{D}^*) = (0.52+0.53)/2 = 0.53,$$

$$E(\mathbf{V};\mathbf{D}^*) = E_{0.5}(\mathbf{V};\mathbf{D}^*) = (0.20+0.14)/2 = 0.17.$$

For the two candidates $\mathbf{U}$ and $\mathbf{V}$, my expectation is correct: E clearly favors $\mathbf{U}$ to $\mathbf{V}$, that is, $\mathbf{U}_{2^*} = \mathbf{U}$.

The objective function $E(\mathbf{U};\mathbf{D}^*)$ is always valued in $[0,1]$. $E(\mathbf{U};\mathbf{D}^*) = 0$ if and only if $I(\mathbf{D}^*)$ has only one intensity, which can occur if and only if $\mathbf{D}^*$ has all zero-valued off-diagonal elements. $E(\mathbf{U};\mathbf{D}^*) = 1$ if and only if $I(\mathbf{D}^*)$ has c perfect (i.e. 0 valued intensities) diagonal blocks with all other off-diagonal intensities equal to one. If the diagonal blocks in Fig. 4.1(a) were pure black, then the partition in Fig. 4.1(b) would result in $E(\mathbf{U};\mathbf{D}^*) = 1$. Remark 3.1.1 showed that the finite set of all aligned partitions $\mathbf{M}^*_{hcn}$ is much smaller than the finite set of all crisp partitions $\mathbf{M}_{hcn}$; however, $\mathbf{M}^*_{hcn}$ is still far to big for an exhaustive search. This leads us to methods for approximating a solution to Eq.(4.6).

4.1.1 Particle Swarm Optimization and CLODD

I stress that, in principle, any number of optimization algorithms could be used. I use *particle swarm optimization* (PSO, [Clerc and Kennedy, 2002]) because it is simple, and because it has been shown to be relatively successful at optimizing highly modal non-linear objective functions. For a given c in $\mathbf{C}$, each candidate $\mathbf{U} = \{n_1 : \ldots : n_c\} \in \mathbf{M}^*_{hcn}$ is completely specified by the c integer indices $\{n_1 : \ldots : n_c\}$, which in turn can be used to specify the locations along the columns of $\mathbf{D}^*$ where trial boundaries are matched to the boundaries in $\mathbf{D}^*$. The integers $m_j = \sum_{k=1}^{j} n_k, j = 1, 2, \ldots, t-1$, are the locations of the right edges (boundaries) of the first $t-1$ blocks in $\mathbf{D}^*$ — the right edge of the last block is at location $m_n = n$, which is the right edge of the matrix or image of the matrix. Because one can recover the c integer $\{n_i\}$ from the $c-1$ integers $\{m_i\}$, I write $\mathbf{U} = \{n_1 : \ldots : n_c\} = \vec{m} \in \mathbf{M}^*_{hcn}$. The vector $\vec{m} = (m_1, \ldots, m_{t-1}) \in \mathbb{R}^{t-1}$ plays a central role in CLODD.

Fix $c = t$. Let $\mathbf{U}_{it} = \{n_{i1} : \ldots : n_{it}\} \in \mathbf{M}^*_{htn}$. Construct the vector $\vec{m}_{it} = (m_{i1}, \ldots, m_{i(t-1)}) \in \mathbb{R}^{t-1}$. This vector of $t-1$ integers has strictly increasing components, $m_{i1} < m_{i2} < \ldots < m_{i(t-1)}$, that specify the $t-1$ locations of the interior boundaries imposed on $\mathbf{D}^*$ by $\mathbf{U}_{it}$. The vector $\vec{m}_{it}$ is thought of as a particle having velocity $\vec{v}_{it} = (v_{i1}, \ldots, v_{i(t-1)}) \in \mathbb{R}^{t-1}$. Let N_p be the number of particles in each swarm, or the number of trial partitions of $\mathbf{O}$ per iteration. Each swarm represents a different choice of the number of clusters t. Let $\hat{m}_{it}$ denote the current best position of each particle in swarm t, let $\hat{\hat{m}}_t$ denote the current best position of all N_p particles in swarm t, and let $\hat{\hat{G}}$ be the best particle over all swarms. In the specification, $\mathbf{rand}([a, b])$ is a random vector, where each component distributed is uniformly on $[a, b]$. With these conventions, I am ready to state the CLODD algorithm, displayed in Algorithm 4.1.1.

Although the CLODD algorithm looks complex, it is really quite simple. Line 2

Algorithm 4.1.1: CLODD: Extraction of clusters from ordered dissimilarity data

```
Input: An $n \times n$ matrix of *ordered* (from, e.g., VAT) dissimilarities,
$\mathbf{D}^* = [d^*_{ij}]; \forall i, j : 0 \leq d^*_{ij} \leq 1, d^*_{ij} = d^*_{ji}, d^*_{ii} = 0$.
Parameters:
$\mathbf{C} = \{2, 3, \ldots, c_{max}\}$ = range of values for search over $\mathbf{M}^*_{hcn}$
$N_p$ = no. of particles for each swarm $c \in \mathbf{C}$
$\alpha$ = mixing coefficient for $E_\alpha(\mathbf{U}; \mathbf{D}^*), 0 \leq \alpha \leq 1$
$\gamma$ = set point control for $S_\gamma(\mathbf{U}), 2/n < \gamma \leq 1$
$q_{max}$ = maximum number of swarm iterations
$\epsilon$ = threshold multiplier
$\epsilon_c = \epsilon N_p(c-1), c \in \mathbf{C}$ = termination threshold at each value of $c$
PSO parameters:
$K$ = inertial constant, $0 < K < 1$
$A_{local}$ = local influence constant, $0 < A_{local} < 4$
$A_{global}$ = global influence constant, $0 < A_{global} < 4$
Main Loop:
1  for $t = 2$ to $c_{max}$ do
2      Initialize particles, $(i, t)$, $i = 1, 2, \ldots, N_p$
3      for $q = 1$ to $q_{max}$ do
4          for $i = 1$ to $N_p$ do
5              if $\vec{m}^{(q)}_{it}$ produces a valid partition then
6                  Build the partition matrices $\mathbf{U}^{(q)}_{it}, \hat{\mathbf{U}}_{it}$, and $\hat{\hat{\mathbf{U}}}_t$ equivalent to
                   $\vec{m}^{(q)}_{it}, \hat{m}_{it}, \hat{\hat{m}}_t$
7                  if $E(\mathbf{U}^{(q)}_{it}) > E(\hat{\mathbf{U}}_{it})$ then $\hat{m}_{it} = \vec{m}^{(q)}_{it}$
8                  if $E(\mathbf{U}^{(q)}_{it}) > E(\hat{\hat{\mathbf{U}}}_t)$ then $\hat{\hat{m}}_t = \vec{m}^{(q)}_{it}$
9              $\vec{v}^{(q+1)}_{it} = K\vec{v}^{(q)}_{it} + A_{local} \cdot \mathbf{rand}([0,1])(.*)(\hat{m}_{it} - \vec{m}^{(q)}_{it}) + A_{global} \cdot$
               $\mathbf{rand}([0,1])(.*)(\hat{\hat{m}}_t - \vec{m}^{(q)}_{it})$
10             $\vec{m}^{(q+1)}_{it} = \text{Round}(\vec{m}^{(q)}_{it} + \vec{v}^{(q+1)}_{it})$
11             CLIP $\vec{m}^{(q+1)}_{it}$, constrain the elements of $\vec{m}^{(q+1)}_{it}$ to the interval
               $[1, n-1]$
12             SORT $\vec{m}^{(q+1)}_{it}$, such that $m_{(it)_1} \leq m_{(it)_2} \leq \ldots \leq m_{(it)_{t-1}}$
13         if $\left[\sum_{s=1}^{t-1}\sum_{i=1}^{N_p} |\vec{v}^{(q+1)}_{is}| < \epsilon_t = \epsilon N_p(t-1)\ \mathbf{OR}\ q = q_{max}\right]$ then STOP
14     if $E(\hat{\hat{\mathbf{U}}}_t) > E(\mathbf{U}_{\hat{\hat{G}}})$ then $\hat{\hat{G}} = \hat{\hat{m}}_t$
```

initializes the particles according to the following procedure:

1. Randomly choose $\vec{m}_{it}^{(1)}$ so that

$$\vec{m}_{it}^{(1)} \neq \vec{m}_{st}^{(1)}, i \neq s,$$

   and

$$\vec{m}_{it}^{(1)} \leftarrow \mathbf{U}_{it}^{(1)} \in \mathbf{M}_{htn}^{*}.$$

2. $\vec{v}_{it}^{(1)} = \mathbf{rand}([-1,1]) : \hat{m}_{it} = \vec{m}_{it}^{(1)} : \hat{\hat{m}}_{t} = \vec{m}_{1t}^{(1)}$

Line 6 builds the candidate partitions according to the particles, including the particles' current location, prevsiou best personal location, and previous best overall location. Although I show in the CLODD algorithm outline that candidate partitions are built at every iteration of the particle swarm, because this problem is discrete in nature, candidate partitions *only* need to built and tested when new candidates are explored. If a candidate partition has been tested in a previous iteration, the objective function does not need to be calculated again. Lines 7 and 8 test to see if candidate partitions are better than the best previously found candidate partitions. Line 9 is the PSO update equation, which updates the velocity of each particle. Line 10 calculates the new location of each particle. Lines 11 and 12 are of particular interest and lead to the following remark.

Remark 4.1.1. It is possible that at the end of the Round operation, $\vec{m}^{(q+1)}$ could have one or more negative entries. This would be not be a valid partition. For example, one might have $\vec{m}^{(q+1)} = (-2,-1,0,3,1)$ *before* clipping. This condition is only temporary, because $\vec{m}^{(q+1)}$ is clipped before it has a chance to reach the objective function. Thus, CLIP $(-2,-1,0,3,1) = (1,1,1,3,1)$. In this example, there are several equal elements in the clipped $\vec{m}^{(q+1)}$. This is *NOT* a valid partition, because it violates the condition that $m_1 < m_2 < \ldots < m_{t-1}$. When this occurs, CLODD will not evaluate the objective function and, subsequently, will not update the local or best particle positions. The particle is allowed to stay in its location (which is invalid) but does not contribute. If the particle is lucky, it will be updated to a *valid* location at the next iteration.

Line 12 sorts the elements of $\vec{m}^{(q+1)}$ such that the elements are ordered and increasing. Note that this is merely a ease-of-computation step and does not change the partition, as the elements of $\vec{m}$ are the boundaries of the aligned partition on the image. Line 13 is the termination criterion for the PSO.

Remark 4.1.2. If the termination criterion $\sum_{s=1}^{t-1}\sum_{i=1}^{N_p}|\vec{v}_{is}^{(q+1)}| < \epsilon_t = \epsilon N_p(t-1)$ is met, the average value of the magnitude of the particle velocities is less than ϵ. There are $(t-1)$ velocity elements in each particle. The particles can only move in discrete jumps (integers, see line 10); hence, an average velocity less than $\epsilon = 0.5$ virtually ensures that all particles have converged to a solution—usually, but not necessarily, the globally best solution of Eq.(4.6).

Finally, line 14 keeps track of the best candidate partition over all values of t, the number of clusters.

Remark 4.1.3. Two or more particles can occupy the same location. In fact, as a swarm approaches termination by the velocity criterion, many particles may be located at the global maximum. As a result of the formulation of the update equation (line 9), once the particles arrive at the global maximum (with minimal momentum), they stay.

Remark 4.1.4. The VAT image of $\mathbf{D}$ may suggest that the data set it represents contains c clusters, but it is CLODD that extracts an aligned c-partition of $\mathbf{O}^*$ from the image. Proposition 3.2.1 shows that when the MST of $\mathbf{D}$ is unique, CLODD extracts SL clusters from the VAT image. The important point is that CLODD extracts only the aligned SL partition at the (VAT suggested) "best value" for c, whereas SL produces $n-1$ aligned c-partitions of $\mathbf{O}$. Thus, [VAT + CLODD] is, for some (but not all) $\mathbf{D}$'s that have a unique MST, equivalent to [SL + some heuristic means for choosing the "best" SL partition].

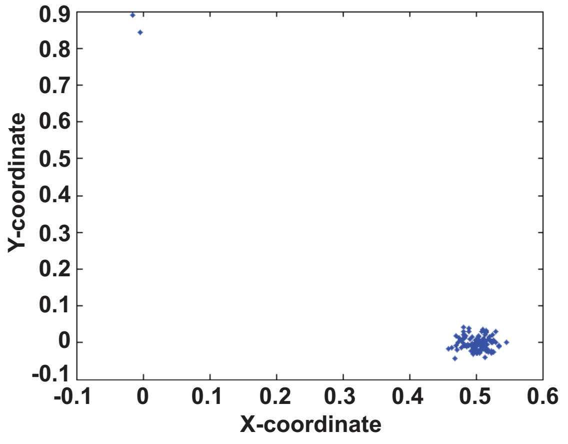
4.1.2 Numerical Examples

This section contains a number of examples that illustrate various facets of the CLODD algorithm. First, I list the computing protocols (for all examples except where noted). $\mathbf{C} = \{2, 3, \ldots, c_{max}\}$ varies from example to example; $\alpha = 0.5$, $\gamma = 0.05$; $N_p = 20$ particles per swarm; $q_{max} = 1000$; $\epsilon = 0.5 =$ termination threshold multiplier; $K = 0.75$; $A_{local} = A_{global} = 2$. Many papers attempt to establish "best" choices for the PSO parameters. I chose the values shown after a limited amount of experimentation with each. A given problem may warrant other choices, but here, I concentrate on the showing the basic points of CLODD.

CLODD parametric evaluation

Figure 4.5(a) shows $n = 100$ object vectors $\mathbf{X} \subset \mathbb{R}^2$. Call the subset of $n_1 = 2$ points in the upper left corner of this scatter-plot $\mathbf{X}_1$, and the remaining $n_2 = 98$ points $\mathbf{X}_2$. The visually apparent cluster structure is the 2-partition $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2$, so I expect $\mathbf{U}_2^* \leftrightarrow \{2 : 98\} \in \mathbf{M}_{h2n}^*$. Convert $\mathbf{X}$ to dissimilarity data $\mathbf{D}$ by calculating the Euclidean distances $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$ for $1 \leq i, j \leq n$.

Figure 4.5(b) shows the VAT image $I(\mathbf{D}^*)$ of $\mathbf{D}$. The structure seen in the scatter-plot is mirrored exactly by $I(\mathbf{D}^*)$ for these nice data. Choose $\mathbf{C} = \{2, 3, 4, 5, 6\}$. First I set $\alpha = 0.5$, and vary the parameter γ, $\gamma = 0.02, 0.05, 0.10, 0.25, 0.50$, and 0.75.



(a) $\{2:98\}$ Object Data

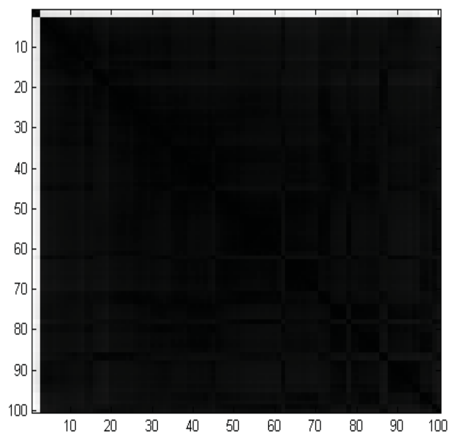
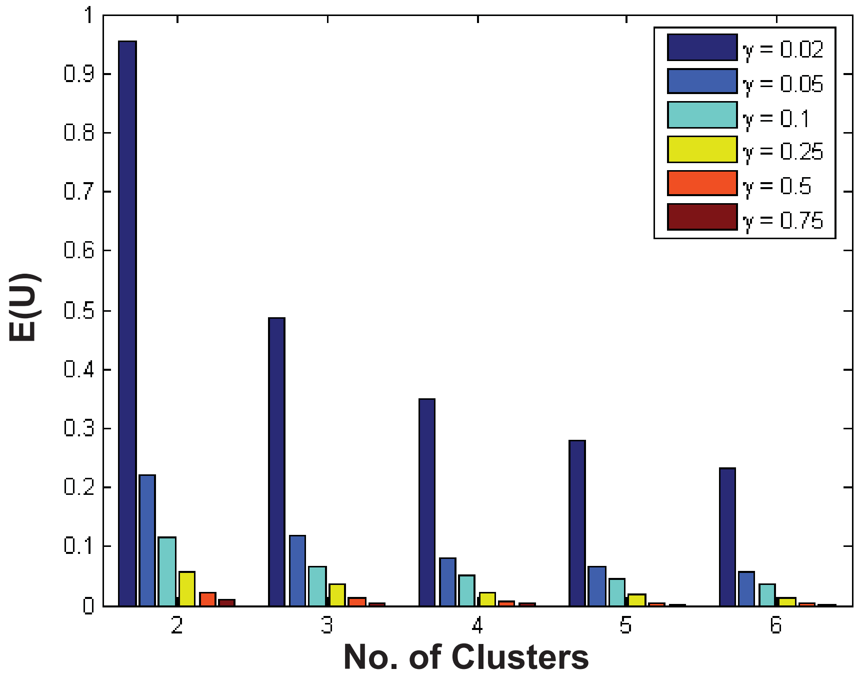

(b) VAT Image $I(\mathbf{D}^*)$

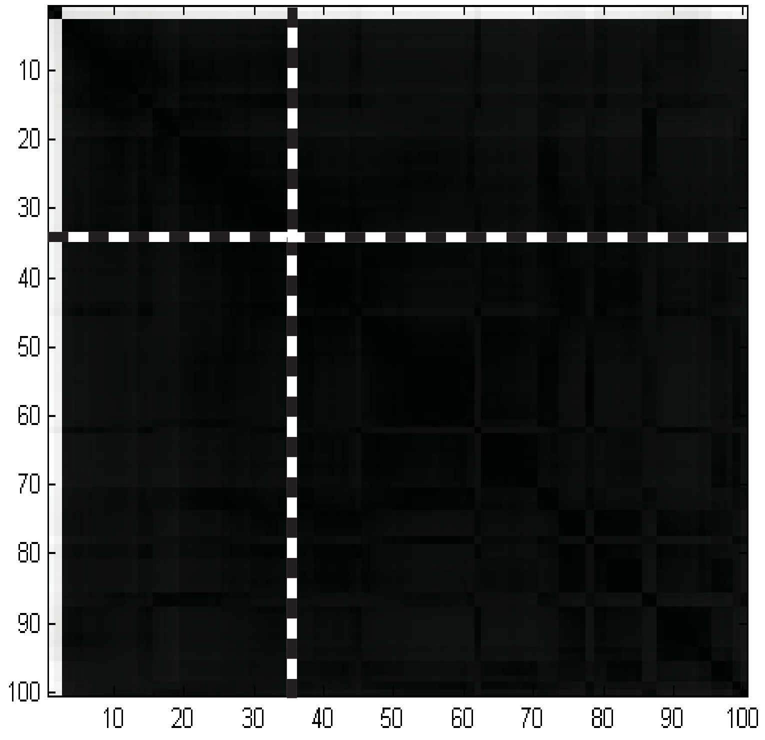
Figure 4.5: Scatter-plot and VAT image of $\{2:98\}$ object data

Figure 4.6 shows the values of $E(\mathbf{U}_c;\mathbf{D}^*)$ for the winner of the PSO competition at each c in $\mathbf{C}$ as γ runs across the specified range of values.

Figure 4.6(a) shows that γ makes an important contribution to the solution of Eq.(4.7). $E(\mathbf{U}_2;\mathbf{D}^*)$ has a strong maximum across all values of c, but as γ increases, the optimal partition at $c=2$ selected by PSO becomes less and less attractive (remember, $E(\mathbf{U};\mathbf{D}^*)$ is maximized, and its value always lies in $[0,1]$). For the parameters of this example, the optimal partition of $\mathbf{D}^*$ (and, hence, of $\mathbf{D}$, and thus $\mathbf{X}$) occurs at $\gamma=0.02$ and $c=2$, for which $\mathbf{U}_{2*}=\{2:98\}\in\mathbf{M}^*_{h2n}$. The first two columns of Table 4.1 show resubstitution error rates, $E_r=100(\#\text{ wrong})/100$ for this set of experiments. Even though $c=2$ wins the PSO competition for every value of γ, only $\gamma=0.02$ realizes zero errors — i.e., finds the "right" 2-partition. For example, $\mathbf{U}_2=\{34:66\}$ for $(\alpha,\gamma)=(0.5,0.75)$. This (wrong) solution is shown in Fig. 4.6(b), where the boundaries of the aligned partition $\{34:66\}$ are shown as an overlay on the VAT image of the data. This misfit is clear in Fig. 4.6(b): 2 of the 34 points in the first cluster are correct, but the remaining 32 points assigned to cluster 1 should have been labeled to cluster 2, so there is a 32% labeling error.



(a) γ vs. c: $\alpha = 0.5$



(b) $\mathbf{U}_{2*}$ for $(\alpha, \gamma) = (0.5, 0.75)$: $c = 2$

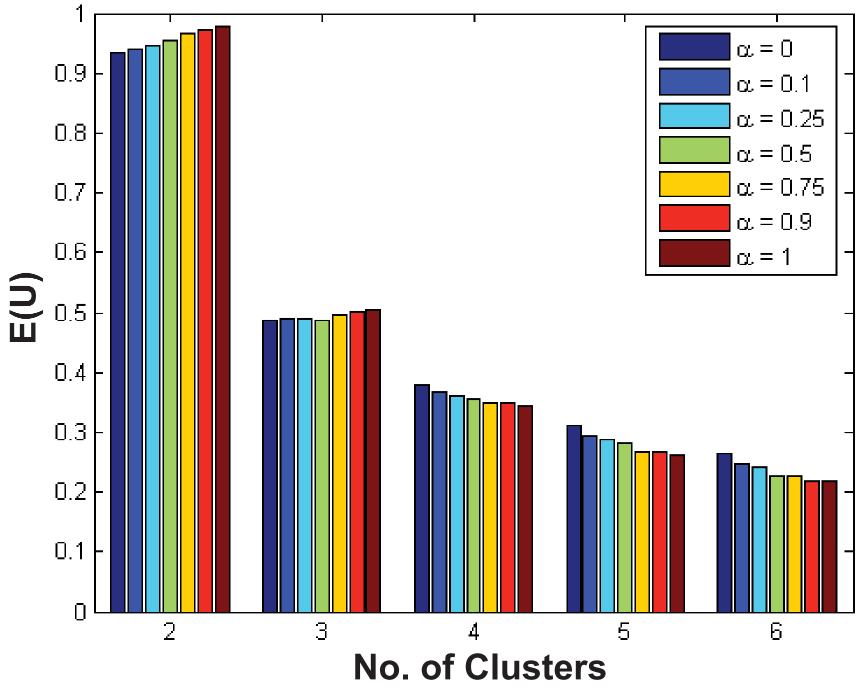
Figure 4.6: The effect of γ on the behavior of CLODD for the $\{2 : 98\}$ data. (a) shows that the 2-partition produces the optimum $E(\mathbf{U}; \mathbf{D}^*)$ at each γ. (b) shows the associated 2-partition for $(\gamma, \alpha) = (0.5, 0.75)$. Note that $\gamma = 0.75$ produces a behavior in CLODD that prefers a partition that has a larger cluster (by design).

Table 4.1: Resubstitution Error Rates for Selected (α, γ) Pairs

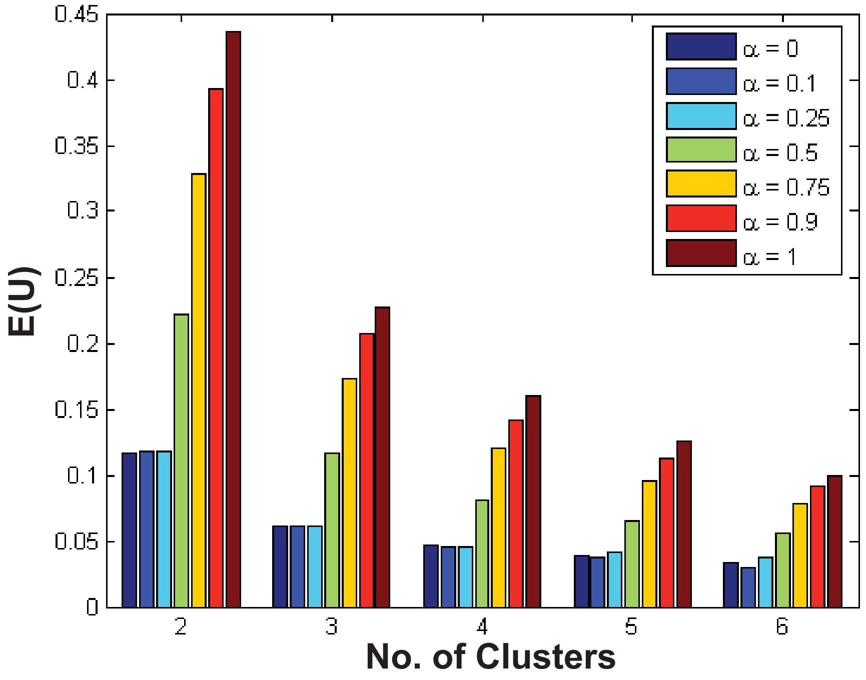
$\alpha = \mathbf{0.50}$		$\gamma = \mathbf{0.02}$		$\gamma = \mathbf{0.05}$	
γ	$\%E_r$	α	$\%E_r$	α	$\%E_r$
0.02	0	0	0	0	0
0.05	2	0.1	0	0.1	0
0.1	6	0.25	0	0.25	0
0.25	17	0.5	0	0.5	2
0.75	32	0.75	0	0.75	2
		0.9	0	0.9	2
		1	0	1	2

Now fix $\gamma = 0.02$, and vary the parameter α, $\alpha = 0, 0.10, 0.25, 0.50, 0.75, 0.90$, and 1.0. Recall that $\alpha = 0$ causes CLODD to *only* use the edge detection of Eq.(4.2, while $\alpha = 1$ causes CLODD to *only* use the block detection of Eq.(4.1). The middle two columns of Table 4.1 show that at $\gamma = 0.02$ every value of the mixing coefficient α produces a perfect fit, $\mathbf{U}_{2*} = \{2 : 98\}$. Figure 4.7 is a plot of the winning values of $E(\mathbf{U}; \mathbf{D}^*)$. At $c = 2$ the objective function shows a slight increase as the parameter α increases from 0 to 1, but this trend does not hold for $c > 2$. Recall that $E(\mathbf{U}; \mathbf{D}^*)$ is bounded above by 1, and notice that the range of values for $E(\mathbf{U}_2; \mathbf{D}^*)$ at $c = 2$ is 0.94 to 0.98. The magnitude of these values provides very strong evidence for the correctness of the fit of $\mathbf{U}_{2*} = \{2 : 98\}$ to $\mathbf{D}^*$. CLODD works well with any choice of α because at $\gamma = 0.02$, the spline function in Eq.(4.5) only penalizes partition choices with clusters smaller than $\gamma n = 0.02(100) = 2$.

The last two columns in Table 4.1 and the plots in Fig. 4.7(b) show the results of repeating this experiment with $\gamma = 0.05$ (penalizing partition choices with clusters smaller than 5 objects). First, notice that the vertical scale in Fig. 4.7(b) is less than half of the scale in view 4.7(a), so every partition recommended by the PSO for $\gamma = 0.05$ is much less attractive (in the eyes of CLODD) than those found with the samller value of γ. Second, observe that the best value of $E(\mathbf{U}; \mathbf{D}^*)$ increases



(a) α vs. c: $\gamma = 0.02$



(b) *alpha* vs. c: $\gamma = 0.05$

Figure 4.7: The effect of α on the value of $E(\mathbf{U};\mathbf{D}^*)$ for the $\{2:98\}$ object data. For each α in both (a) and (b) the 2-partition is the preferred partition. For $\alpha = 1$, $E(\mathbf{U};\mathbf{D}^*) = S_\gamma(\mathbf{U}) \cdot E_{\text{sq}}(\mathbf{U};\mathbf{D}^*)$, and for $\alpha = 0$, $E(\mathbf{U};\mathbf{D}^*) = S_\gamma(\mathbf{U}) \cdot E_{\text{edge}}(\mathbf{U};\mathbf{D}^*)$.

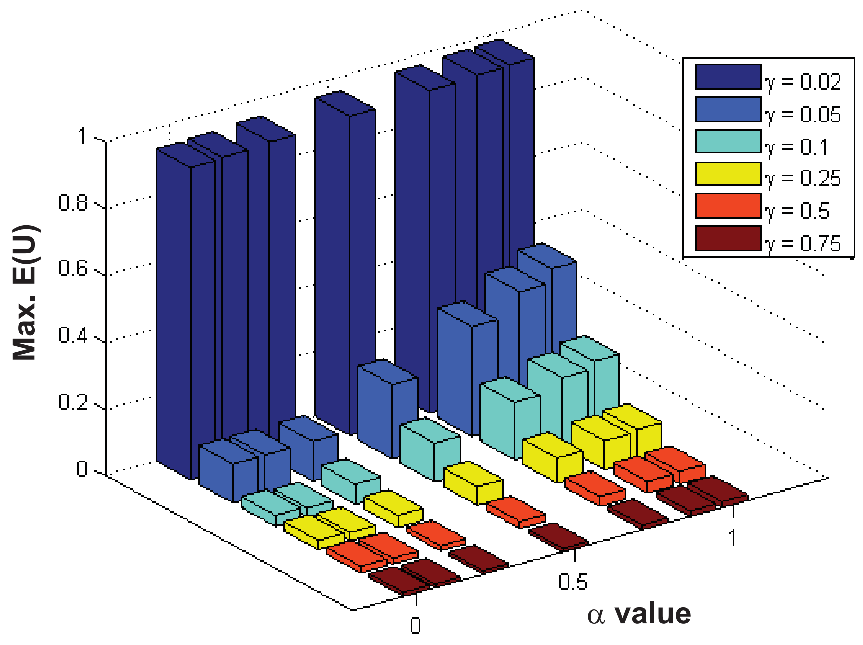
Table 4.2: Resubstitution Error Rates for Selected (α, γ) Pairs At $c = 2$ For The $\{2 : 98\}$ Data

	γ					
α	0.02	0.05	0.10	0.25	0.50	0.75
0	0	0	85	75	59	59
0.10	0	0	85	17	59	59
0.25	0	0	6	17	43	43
0.50	0	2	6	17	27	32
0.75	0	2	6	17	27	32
0.90	0	2	6	17	27	32
1	0	2	6	17	27	32

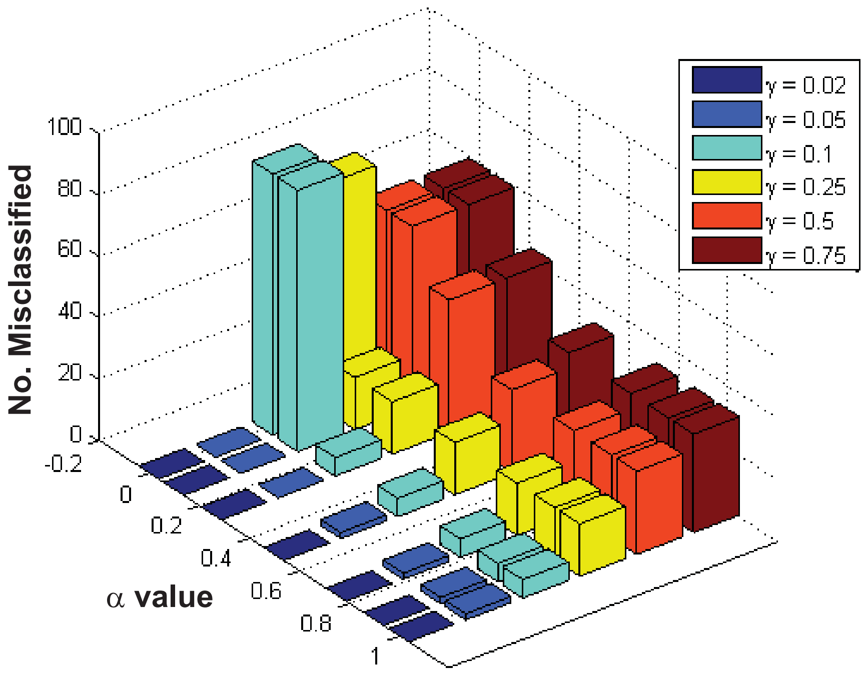
monotonically as α increases from 0 to 1, but the largest value in Fig. 4.7(b) is less than half of the best value in view 4.7(a). Third, Table 4.1 shows that the desired best match partition, $\mathbf{U}_{2*} = \{2 : 98\}$ is found when $\alpha = 0, 0.10$, and 0.25, but for all higher value of α — which occur at higher value of E — the partition chosen is $\mathbf{U}_{2*} = \{4 : 96\}$, a labeling of the objects in $\mathbf{X}$ that has two mistakes. This is a different (and less severe) type of bad behavior than that shown by CLODD when $\alpha = 0.5$ is fixed and γ increases, as shown in the first two columns of Table 4.1.

Figure 4.8(a) shows 42 values of the CLODD objective function $E(\mathbf{U}; \mathbf{D}^*)$ and Fig. 4.8(b) shows the resubstitution error rates corresponding to the PSO winners at $c = 2$. For example, the tall dark bars at the back of Fig. 4.8(a) are the 7 bars in the leftmost group at $c = 2$ in Fig. 4.7(a); the zeroes seen in the foreground of Fig. 4.8(b) correspond to the error rates shown in the middle two columns of Table 4.1.

Table 4.2 shows the data that are plotted in Fig. 4.8(b). The largest number of misclassifications, 85%, occurs at two value pairs: $(\alpha, \gamma) = (0, 0.10)$ and $(\alpha, \gamma) = (0.10, 0.10)$. As γ continues to increase, the error drops. In this example, with one very small cluster in these data, CLODD is most reliable when γ is also very small (see Table 4.1). Interestingly, CLODD always chooses $c = 2$; it just gets the wrong 2-partition for large values of γ — this effect is also exacerbated when α is small,

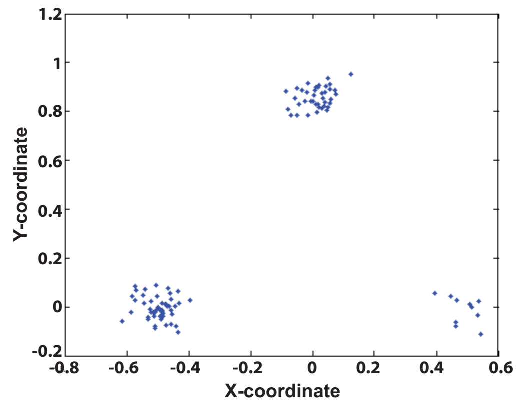


(a) $E(\mathbf{U};\mathbf{D}^*)$ vs. (α,γ)

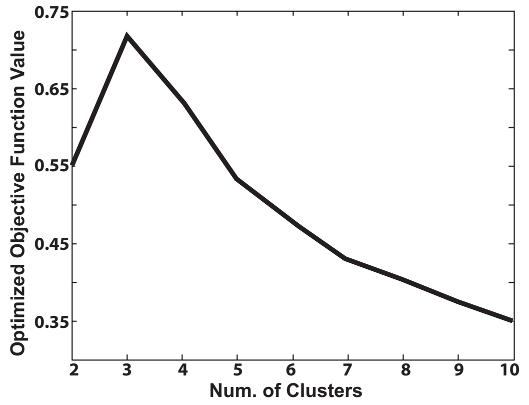


(b) E_r vs. (α,γ)

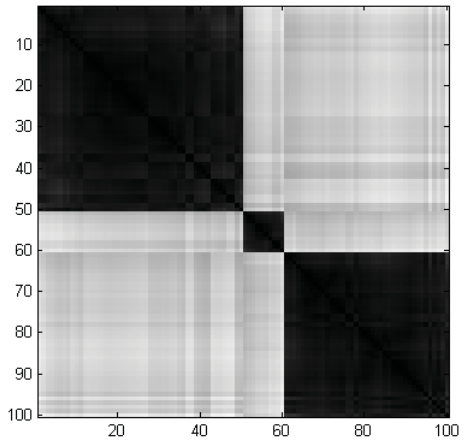
Figure 4.8: Joint distribution of $E(\mathbf{U};\mathbf{D}^*)$ and error rate E_r as a function of (α,γ) for $\{2:98\}$ data: $c=2$. This example shows that $\gamma=0.02$ produces the error rate.



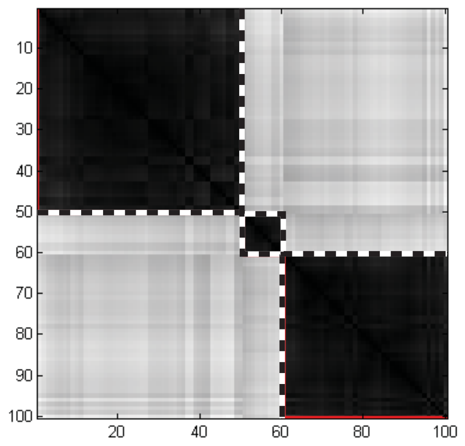
(a) Three clouds data $\mathbf{X}_3$



(b) $E(\mathbf{U}_c; \mathbf{D}_3^*)$ vs. c



(c) VAT image $I(\mathbf{D}_3^*)$

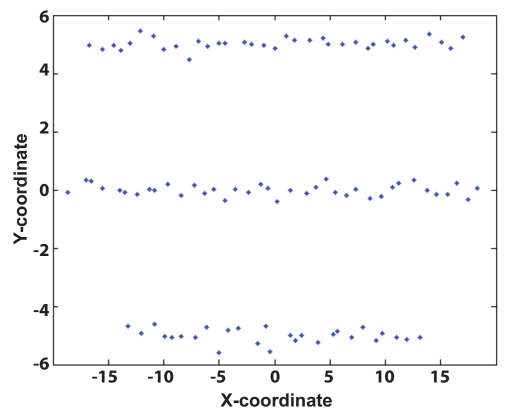


(d) VAT image with $\mathbf{U}_{3*}$

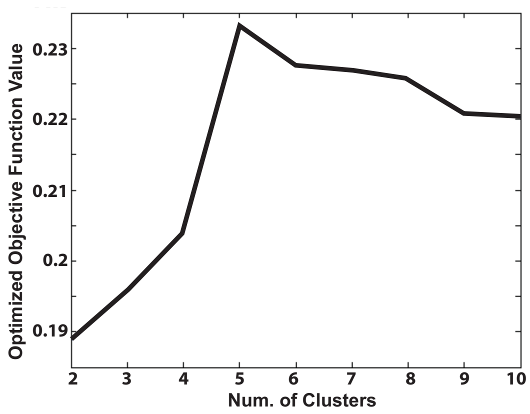
Figure 4.9: Object data scatter-plot, PSO winners, VAT image, and optimal CLODD partition for the Three Clouds data $\mathbf{X}_3$ — dotted line in view (d) indicates partition boundaries. CLODD found the preferred 3-partition.

meaning the objective function is looking for edginess. All of the remaining examples use the fixed values $(\alpha, \gamma) = (0.50, 0.05)$. These values were empirically chosen as a good trade-off between edge-detection and block-detection, and also penalizing small cluster sizes.

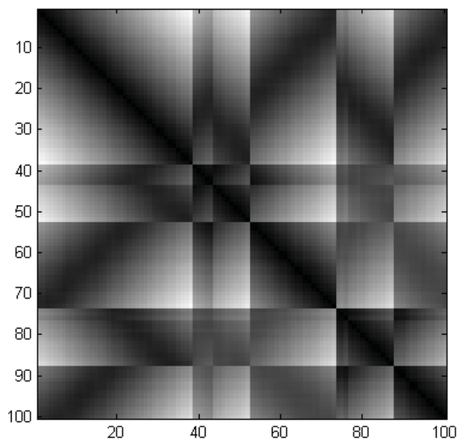
Example 4.1.2 (Three Gaussian Clouds). Figure 4.9(a) shows $n = 100$ object vectors $\mathbf{X}_3 \subset \mathbb{R}^2$. Figure 4.9(c) is the VAT image $I(\mathbf{D}_3^*)$ of the corresponding Euclidean dissimilarity data $\mathbf{D}_3$. The well-defined cluster structure that is visually evident in $\mathbf{X}_3$ is represented exactly in $I(\mathbf{D}_3^*)$, so CLODD is expected to find a perfect match



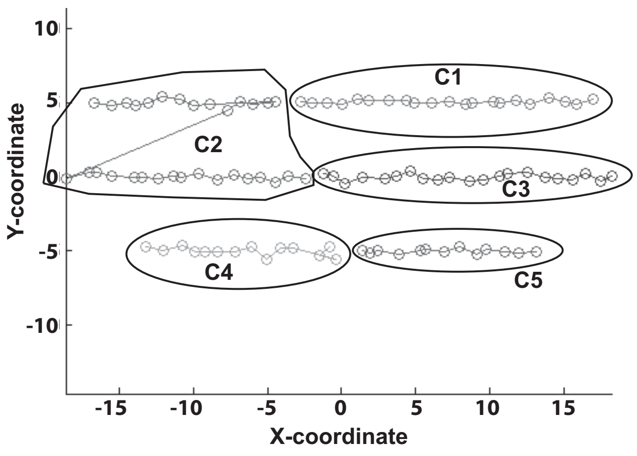
(a) Three lines data $\mathbf{X}_3$



(b) $E(\mathbf{U}_c; \mathbf{D}^*_{3L})$ vs. c



(c) VAT image $I(\mathbf{D}^*_{3L})$



(d) CLODD solution U_{5*}

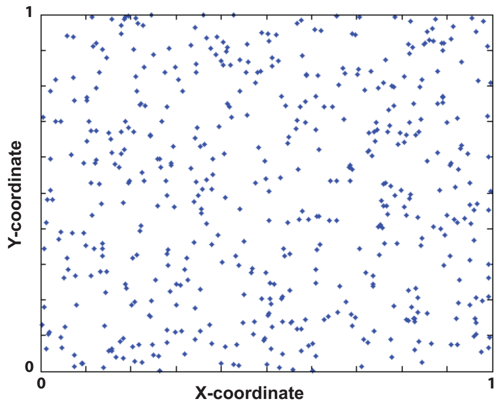
Figure 4.10: VAT image, PSO winners, and optimal CLODD partition for the 3 lines data $\mathbf{X}_{3L}$. CLODD was unable to find the preferred 3-partition.

to the boundaries in the VAT image. Figure 4.9(b) is a plot of the values of the objective function $E(\mathbf{U}_c; \mathbf{D}^*_3)$ for the PSO winners at each $c = 2, 3, \ldots, 10$. The aligned partition $\mathbf{U}_{3*}$ has a strong maximum of 0.72 in Fig. 4.9(b). This partition — the expected perfect match — is superimposed on $I(\mathbf{D}^*_3)$ in Fig. 4.9(d).

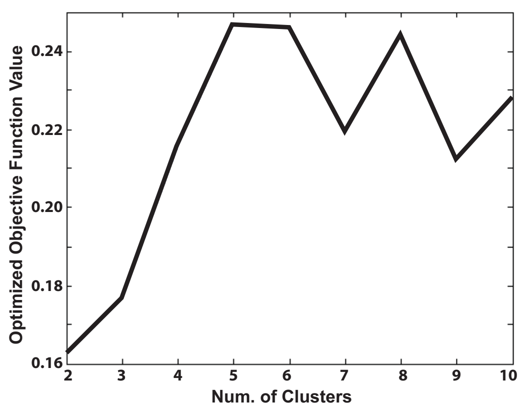
Example 4.1.3 (Three lines). Figure 4.10(a) shows $n = 100$ object vectors $\mathbf{X}_{3L} \subset \mathbb{R}^2$. Figure 4.10(c) is the VAT image $I(\mathbf{D}^*_{3L}$ of the corresponding Euclidean dissimilarity data $\mathbf{D}_{3L}$. Most observers would agree that there is well defined cluster structure, which is visually evident in $\mathbf{X}_{3L}$, but view 4.10(c) shows that VAT does not elicit this from these data. The visual impression given by $I(\mathbf{D}_{3L}^*$ is that $\mathbf{X}_{3L}$ has $c = 5$ clusters, and it is seen that CLODD agrees. The PSO winners at each c, shown in Fig. 4.10(b), have a clear maximum at $c = 5$. Please notice that the corresponding aligned partition $\mathbf{U}_{5*}$, which solves Eq.(4.7), has a very weak maximum of 0.233. This partition of $\mathbf{X}_{3L}$ is shown in Fig. 4.10(d). What went wrong? VAT failed to reorder the distance matrix to show the $c = 3$ linear clusters. As shown in Section 3.3, the ability of VAT to show "proper" cluster tendency is directly linked to Dunn's cluster validity index. Dunn's index for the visually apparent 3-partition of $\mathbf{X}_{3L}$ is ˜0.3, which is less than 1; hence, the contrast of the VAT image is not sufficient to show a cluster tendency of $c = 3$.

Example 4.1.4 (Uniform random field). To study the candidate partitions that CLODD might suggest when there are *no* visible clusters in the data, I generated a set of 500 object vectors $\mathbf{X}_u$, uniformly distributed in $[0, 1] \times [0, 1]$, and converted them to Euclidean dissimilarity data $\mathbf{D}_u$. What would you conjecture, based only on the visual evidence in the VAT image $I(\mathbf{D}_u^*)$, shown in Fig. 4.11(c)? There are several dark blocks in the lower part of this image that attract the eye, and there are quite a few smaller dark blocks along the diagonal, so you might speculate that there is some type of cluster substructure in the data — albeit weak and perhaps not distinguishable by the reordering procedure used by VAT.

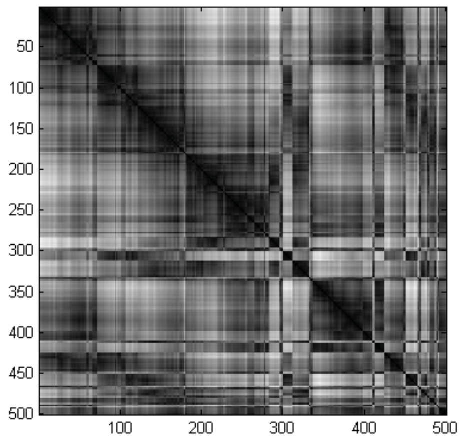
The solution of the CLODD objective function Eq.(4.7) for these data is indicated by the maximum on the graph in Fig. 4.11(b). CLODD finds $c = 5$ clusters, and the corresponding partition is shown in Fig. 4.11(d). The optimal CLODD partition $\mathbf{U}_{5*}$ is not an unreasonable fit to the VAT image. Although, it certainly could be argued that there is *NO* cluster structure in these data. CLODD does not, in essence, fail for this data. CLODD finds an aligned partition that is a pretty good match to the VAT image it has to work with. The failure in this case, as in the Three Lines example,



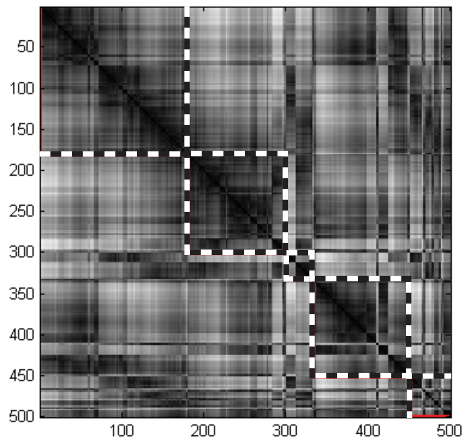
(a) Uniform data $\mathbf{X}_3$



(b) $E(\mathbf{U}_c; \mathbf{D}_u^*)$ vs. c



(c) VAT image $I(\mathbf{D}_u^*)$

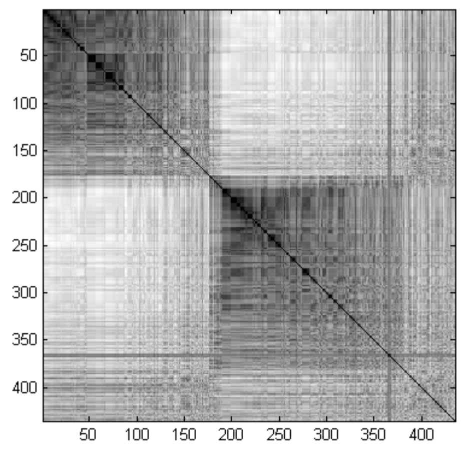


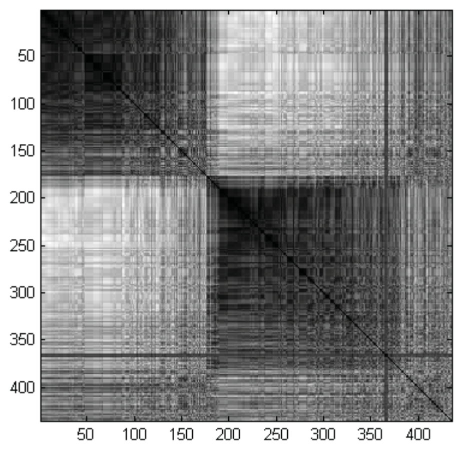
(d) VAT image with $\mathbf{U}_{5*}$

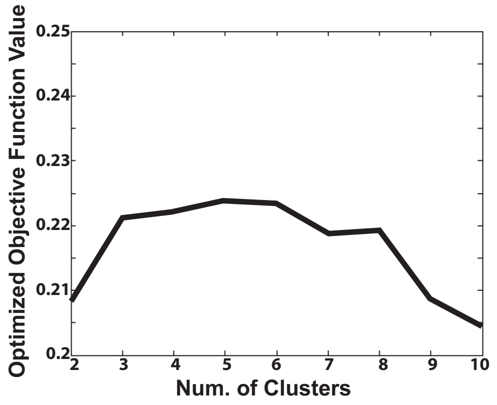
Figure 4.11: VAT image, PSO winners, and optimal CLODD partition for the uniform data $\mathbf{X}_u$ — dotted line in view (d) indicates partition boundaries. There is no preferred partition for this data.

is due to VAT, which produces a reordered image that seems to have more structure than the scatter-plot of these data suggest. This reminds us that the job of every clustering algorithm is to find clusters, and CLODD is not different from all other clustering algorithms in this respect: CLODD does its job — namely, finding clusters where none seem to exist.

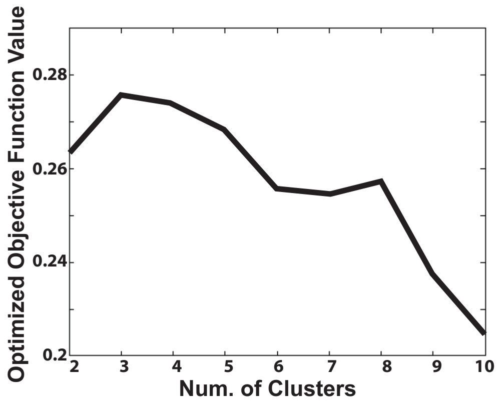
Example 4.1.5 ("VOTE" data). This example uses the real world VOTE data set, downloaded from the UCI Machine Learning Repository [Asuncion and Newman, 2007]. The data are generated from Congressional voting records, and consist of the


(a) VAT image $I(\mathbf{D}_e^*)$

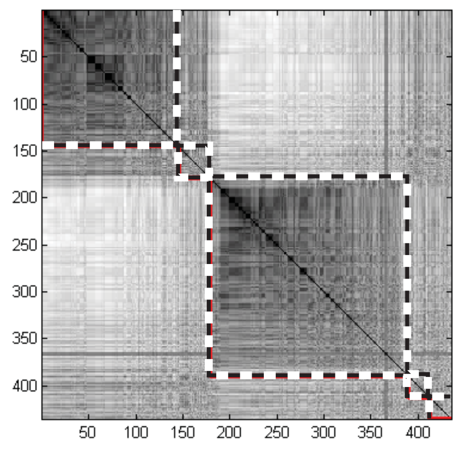

(b) VAT image $I(\mathbf{D}_{e^2}^*)$



(c) $E(\mathbf{U}_c;\mathbf{D}_e^*)$ vs. c



(d) $E(\mathbf{U}_c;\mathbf{D}_{e^2}^*)$ vs. c


(e) $\mathbf{U}_{5*} = \{25 : 24 : 210 : 31 : 145\}$

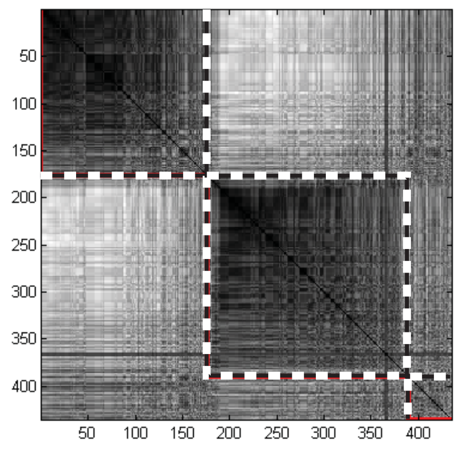

(f) $\mathbf{U}_{3*} = \{176 : 224 : 45\}$

Figure 4.12: VAT images, PSO winners, and optimal CLODD partitions for the VOTE data — dotted line in views (e,f) indicates partition boundaries. Views (a,c,e) use Euclidean dissimilarity relation, a, views (b,d,f) use squared Euclidean dissimilarity relation.

1984 records of the 435 members of the United States House of Representatives on 16 key votes. These data consist of "y" for yea, "n" for nay, and "?" for unknown disposition. To represent these data numerically, I chose the values 0.5 for yea, -0.5 for nay, and 0 for unknown. Thus, the voting records are represented by an object data set $\mathbf{X}_{VOTE} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{435}\} \subset \mathbb{R}^{16}$. Euclidean and squared Euclidean distances were used to generate relational data sets $\mathbf{D}_e$ and $\mathbf{D}_{e^2}$ from $\mathbf{X}_{VOTE}$. Figure 4.12(a) shows the VAT image $I(\mathbf{D}_e^*)$. This image gives the impression that there are two clusters in the data, but the intensities at the edges of the dark regions fade into neighboring pixels more or less continuously, and the lower corner, along the diagonal of the lower block, simply disappears. Figure 4.12(c) plots the values of the objective function for the winner of each PSO competition, where, recall, each PSO competition is for a different number of clusters. The range of values of the vertical axis of Fig. 4.12(c) is very compressed and is relatively small — $E(\mathbf{U}; \mathbf{D}_e^*)$ is valued in $[0.208, 0.223]$. The graph from $c = 3$ to $c = 6$ is nearly flat, so while there is a maximum at $c = 5$, it is relatively weak. This indicates that the optimal CLODD partition $\mathbf{U}_{5*}$ is not *clearly* preferable, just better than those at other values of c.

Figure 4.12(b), the VAT image $I(\mathbf{D}_{e^2}^*)$, has improved visual contrast for the preferred 2-partition along party lines. The dark blocks are darker and the boundaries seem more distinct, but a gray area is still seen along the bottom and right edge of the VAT image. Figure 4.12(d) plots the winning objective function value at each c. Views 4.12(c) and 4.12(d) show that changing the input data from $\mathbf{D}_e$ to $\mathbf{D}_{e^2}$ changes the number of optimal clusters from $c = 5$ to $c = 3$. This demonstrates the ability of the edginess and contrast factors, which comprise $E(\mathbf{U}; \mathbf{D}^*)$, to track changes in contrast and edge definition in the VAT image $I(\mathbf{D}^*)$. The 3-partition chosen as the best match for $I(\mathbf{D}_{e^2}^*)$ is $\mathbf{U}_{3*} = \{176 : 224 : 45\}$. This is a somewhat more satisfying result than the partition $\mathbf{U}_{5*} = \{145 : 31 : 210 : 24 : 25\}$ that CLODD

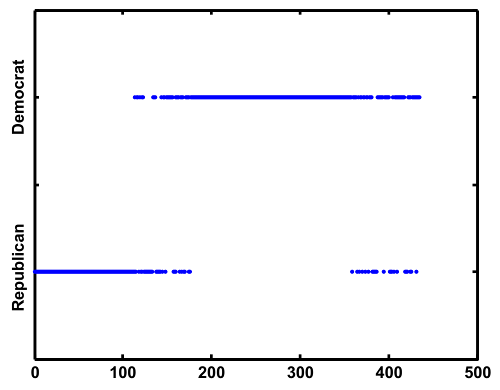
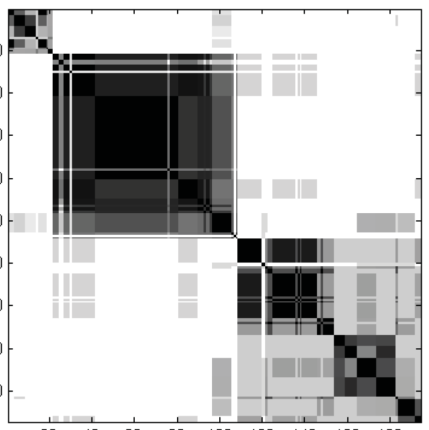


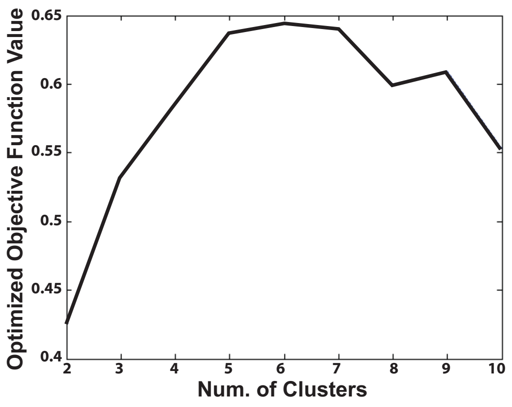
Figure 4.13: Party label of 435 members of congress ordered by VAT.

matches to $I(\mathbf{D}_e^*)$. The two identified classes in these data are Republicans (54.8%) and Democrats (45.2%), but this does not guarantee that the numerical data contain two geometrically well-defined clusters. The two apparent clusters correspond to Democrats and Republicans voting along party lines, while the poorly defined region in the bottom right corner of $I(\mathbf{D}_{e^2}^*)$ corresponds to 45 voters who crossed party lines on these 16 votes. Figure 4.13 shows the party labels of the members in the same order as produced by VAT in Fig. 4.12.

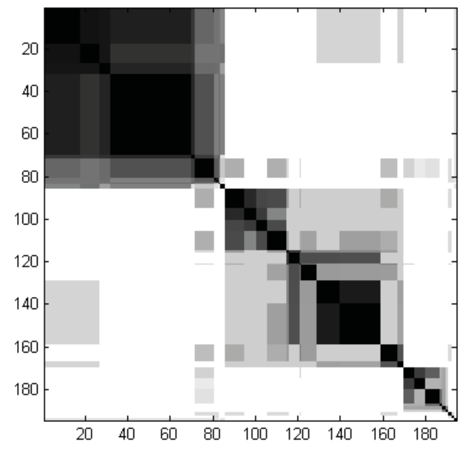
Remark 4.1.5. The contrast enhancement used in the previous example does not change the ordering of the VAT-ordered dissimilarity matrix, it merely improves the visual contrast of the image. However, the contrast enhancement does change the clustering, according to CLODD, because the distance calculation (proximity metric) used is different. This is a strength of CLODD; it is not dependent on a Euclidean distance calculation, it only requires a dissimilarity matrix as input. Because it is a visual technique, one could run CLODD with different contrast enhancement methods and choose the most visually pleasing result.

Example 4.1.6 (Bioinformatics data). The last example uses one version of the real world data $GPD194_{12.10.03}$, denoted here as $\mathbf{D}_{194}$. This data set is also described and shown in Section 2.4 and Figure 2.11. This data is different from the previous


(a) Dissimilarity data D_{194}



(b) $E(\mathbf{U}_c; \mathbf{D}^*_{194}$ vs. c


(c) VAT image $I(\mathbf{D}^*_{194})$

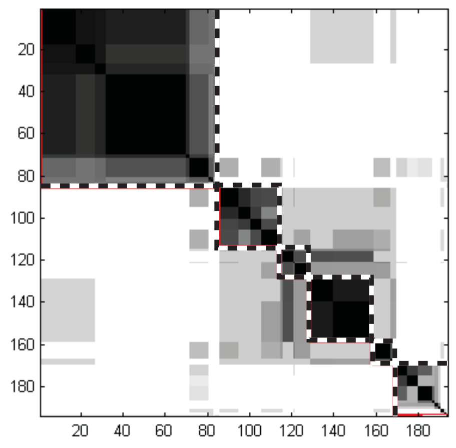

(d) VAT image with $\mathbf{U}_{6*}$

Figure 4.14: VAT image, PSO winners and optimal CLODD partition for the $GPD194_{12.10.03}$ data — dotted line in view (d) indicates partition boundaries. CLODD found substructure within the *collagen* family, which has been reported in other literature.

examples in that it is not derived from object data. Rather, it is derived directly from a (dis)similarity relation built with a fuzzy measure applied to annotations of 194 human gene products, which appear in the Gene Ontology [The Gene Ontology Consortium, 2004]. Popescu et al. [2004] contains a detailed description of the construction of this data and Section 2.4 further describes this data. Recall that these data are comprised of 21 sequences of gene products from the Myotubularin protein family, 87 sequences of gene products from the Receptor Precursor protein family, and

86 sequences of gene products from the Collagen Alpha Chain protein family. The three protein families are clearly visible in the image of $\mathbf{D}_{194}$ shown in Fig. 4.14(a); the upper left block is the Myotubularins, the middle block is the Collagens, and the lower right block is the Receptor Precursors. Note the strong substructure within the Collagen protein family dissimilarity data. This substructure has been corroborated in Myllyharju and Kivirikko [2004] and, as you will see, is also supported by CLODD.

Figure 4.14(a) displays an image of $\mathbf{D}_{194}$, and if you compare this image to the VAT image $I(\mathbf{D}^*_{194})$ in Fig. 4.14(c), you will see that they are similar, but not exactly equal. However, both these images seem to suggest that there are more than just 3 clusters, with $c = 5 - 7$ main clusters being my estimate from the VAT image. In this regard, CLODD agrees. Figure 4.14(b) shows a slight maximum in the objective function at $c = 6$, and the corresponding partition $\mathbf{U}_{6*}$ is shown superimposed in Fig. 4.14(d). In this example, the three highest values of the objective function, which occur at $c =$ 5,6, and 7, are all about 0.64. Compare this to the best values of the objective function in the previous examples. In the $\{2 : 98\}$ and Three Clouds data, the maximum objective function value is larger than 0.6; in both these examples CLODD (arguably) found a good partition of these data. But in the Three Lines, Uniform, and VOTE data, where either VAT or CLODD performed less reliably, the value of the objective function is below 0.25. Hence, I believe that CLODD supports the substructure found in the Collagen family. Also, please note that within the 6 main clusters found by CLODD in the $GPD194_{12.10.03}$ data (which, for lack of a better term, I call *first order* clusters), there are visually apparent subclusters (*second order* clusters).

4.1.3 Perspectives on CLODD

The examples demonstrate that when **D** has "good" clusters, CLODD will find them. In the examples when CLODD finds a good match to a good VAT image of the data, the value of the objective function is larger than 0.6. But in the examples where either VAT or CLODD is less reliable, the value of the objective function is below 0.25. This indicates that CLODD is useful for both finding clusters in unlabeled data and, also, presenting a cluster validity index of those clusters.

There are algorithms besides VAT that produce block diagonal images: some are displays of clusters already found [Johnson and Wichern, 2007, Floodgate and Hayes, 1963, Ling, 1973, Baumgartner et al., 2000, 2001]; others are constructed, like VAT, to assess structure prior to clustering [Baumgartner et al., 2000, Zhang and Chen, 2003]; still others are used to simultaneously find and display clusters [Johnson and Wichern, 2007, Sneath, 1957, Tran-Luu, 1996]; and finally, images with this type of structure are used to attack the validity question [Hathaway and Bezdek, 2003, Huband and Bezdek, 2008]. Consequently, CLODD is much more widely useful than it might appear. However, many good questions remain. For example, the possibility that Eq.(4.7) may not have a solution, or that it has more than one, is ignored. These questions are interesting, but the objective function in Eq.(4.6) is discontinuous on its domain; hence, these questions are indeed formidable.

On a more practical note, I ask if there is a better way than trial and error to find a reliable pair of CLODD parameters (α, γ)? The initial attempts at approaching this question have centered on computational ways to make CLODD "adaptive", but so far, I have met with little success. Another interesting question concerns the reliance of CLODD on VAT. Certainly CLODD will fail when VAT does, and it was demonstrated here that this can happen. It is possible that other reordering methods might be useful "front end" partners for CLODD in such cases. This leads to a related question concerning the size of the data $\mathbf{O}$. VAT is a useful reordering scheme for small to medium sized data sets ($n \leq 10{,}000$). The scalable version of VAT [Hathaway et al., 2006] produces a sample-based estimate of the VAT image $I(\mathbf{D}^*)$ for very large n, but does not reorder the very large data in preparation for CLODD clustering.

4.2 New Formulations of co-VAT

4.2.1 Alternate reordering scheme

The original co-VAT algorithm, outlined in Algorithm 2.3.1, reorders the rectangular matrix $\mathbf{D}$ by shuffling the VAT-reordering indexes of $\mathbf{D}_{r\cup c}$. Thus, co-VAT is very dependent on the construction of $\mathbf{D}_{r\cup c}$. Algorithm 4.2.1 presents a reordering scheme that is not dependent on the reordering of $\mathbf{D}_{r\cup c}$ — this matrix does not even need to be constructed. However, we still need the matrix of the union if we intend to assess cluster tendency in $O_{r\cup c}$. Essentially, the reordering of the row indexes of $\mathbf{D}$ are taken from the VAT-reordering of $\mathbf{D}_r$ and the reordering of the column indexes are taken from the VAT-reordering of $\mathbf{D}_c$. Another advantage of this alternate reordering scheme is that the scale factors, λ_r and λ_c in (2.22) and (2.23), can be ignored.

Algorithm 4.2.1: Alternate co-VAT Reordering Scheme

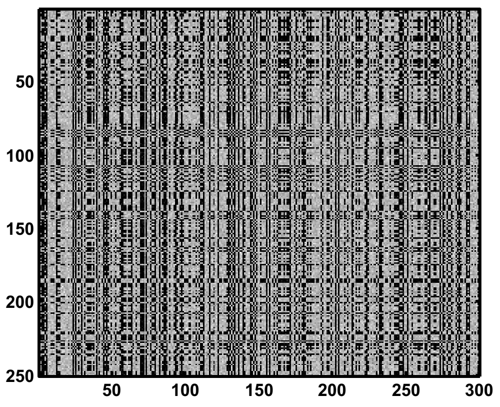
```
  Input: $\mathbf{D}$ - $m \times n$ rectangular dissimilarity matrix
  Build estimates of $\mathbf{D}_r$ and $\mathbf{D}_c$ using Eqs.(2.22) and (2.23), respectively.
1 Run VAT on $\mathbf{D}_r$, saving permutation array, $RP = \{RP(1), \ldots, RP(m)\}$
2 Run VAT on $\mathbf{D}_c$, saving permutation array, $CP = \{CP(1), \ldots, CP(n)\}$
3 Form the co-VAT ordered rectangular dissimilarity matrix,
  $\mathbf{D}^* = [d^*_{ij}] = [d_{RP(i)CP(j)}]$, $1 \leq i \leq m$; $1 \leq j \leq n$
  Output: Rectangular image $I(\mathbf{D}^*)$
```

Appendix A.3 presents several examples that show that the alternate co-VAT is successful for all the cases that co-VAT was shown successful in [Bezdek et al., 2007]. The following two examples illustrate the effectiveness of the alternate co-VAT reordering scheme for data sets where the original co-VAT fails.

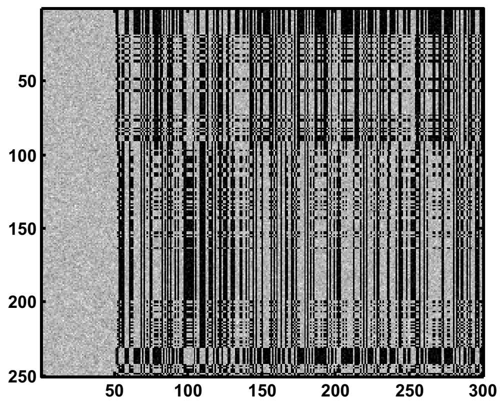
Example 4.2.1. This example is on a pure relational data set that has 250 row objects and 300 column objects. The dissimilarity data for these objects is shown in Fig. 4.15(a). Clearly, view (a) does not indicate any discernable cluster structure. Similarly, the co-VAT image in view (b) also does not show any cluster structure. At the very least, this image shows that there is a group of column objects (indexed 1-50) that do not have a strong similarity to *any* row objects.

Interestingly, views (d) and (e) of $\mathbf{D}_r^*$ and $\mathbf{D}_c^*$, respectively, show the cluster tendency of the row and column objects rather well: 3 clusters in the row objects and 5 clusters in the column objects. But, the reason that co-VAT "fails" on this example is because the reordering of $\mathbf{D}_{r\cup c}^*$ "fails". View (f) shows $\mathbf{D}_{r\cup c}^*$. This image shows no clear cluster structure in the union of the row and column objects. Thus, when the reordering indexes are reshuffled to produce the co-VAT image, this image also fails to show the structure.

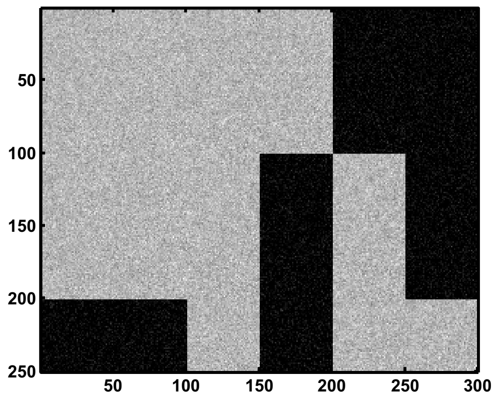
However, if I use the alternate reordering method to produce a co-VAT image, as shown in Fig. 4.15(c), the cluster structure of this data is very clear. This image would suggest 3 pure co-clusters; however, notice that there is overlap between all of the objects that have a strong similarity to another object (notice that the column objects indexed 100-150 are the same column objects indexed 1-50 in the original co-VAT image). The row objects indexed 200-250 have a strong similarity to column objects 1-100 and 150-200. However, the column objects 150-200 are also strongly similar to the row objects 100-200. And these row objects are strongly similar to the column objects 250-300. Thus, I believe that a question remains as to the number of pure co-clusters in this data set. Also, it is my conjecture that these overlaps are the reason that the original co-VAT formulation "fails" for this data set.



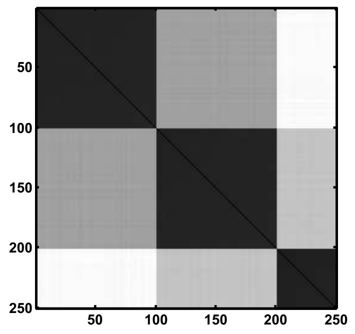
(a) Dissimilarity data



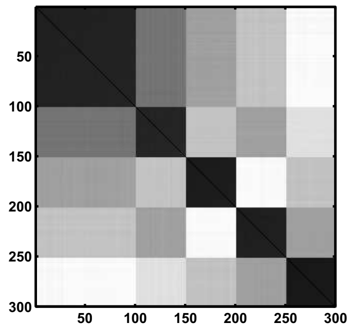
(b) $\mathbf{D}^*$



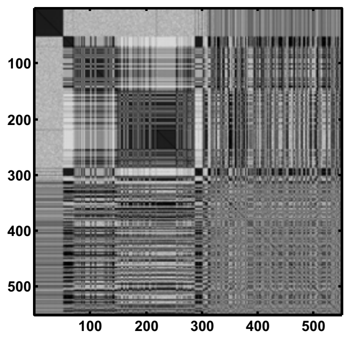
(c) alternate $\mathbf{D}^*$



(d) $\mathbf{D}_r^*$



(e) $\mathbf{D}_c^*$



(f) $\mathbf{D}_{r \cup c}^*$

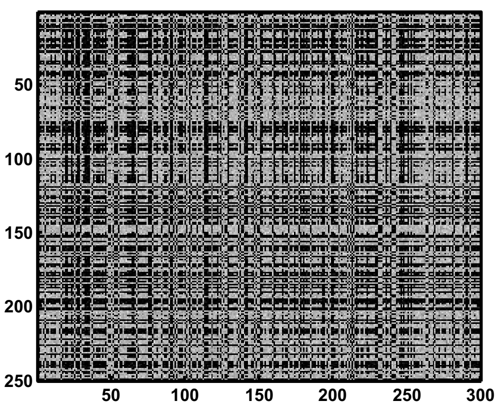
Figure 4.15: Example 4.2.1 Pure rectangular relational data. (a) - dissimilarity data; (b) original co-VAT reordering, (c) alternate co-VAT reordering, (d-f) - co-VAT reordered dissimilarity data matrices.

Example 4.2.2. This example is on a pure relational data set that has 250 row objects and 300 column objects. The dissimilarity data for these objects is shown in Fig. 4.15(a). As in the previous example, the co-VAT image shown in view (b) does not suggest any cluster structure. Again, the images of $\mathbf{D}_r^*$ and $\mathbf{D}_c^*$ clearly show structure: 6 clusters in the row objects and 5 clusters in the column-objects. The image of $\mathbf{D}_{r\cup c}^*$ again shows no clear cluster structure.

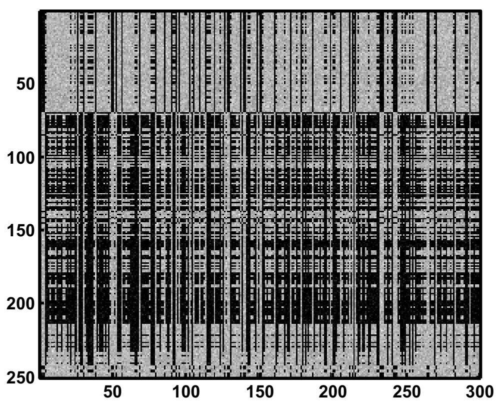
The alternate reordering method, shown in view (c), produces a visually pleasing image. There appears to be 2 pure co-clusters in this data set. However, as we saw in the previous example, there is overlap between the dark blocks—row objects 175-200 are strongly similar to column objects 250-300. Thus, the question remains as to how many pure co-clusters there are. This example also provides more evidence for my conjecture that the overlapping is the cause of the failure of co-VAT.

It is my opinion that in Examples 4.2.1 and 4.2.2 there is 1 pure co-cluster in the union of the row and column objects. Imagine that these dissimilarity matrices represent the weights of a partially-connected graph, where the dissimilarity values represent the edges and objects represent the vertices. Each row vertex is connected to all column vertices and each column vertex is connected to all row vertices. Consider the dissimilarity data shown in Fig. 4.15(c) for Example 4.2.1. It is clear that there exists a "low-cost" path between every set of vertices, except for the column vertices indexed 100-150. Conversely, in Example 4.2.2, the dissimilarity data shown in Fig. 4.16(c) shows that there exists a "low-cost" path between every set of vertices. For this reason, I believe that there is 1 pure co-cluster in these two examples.

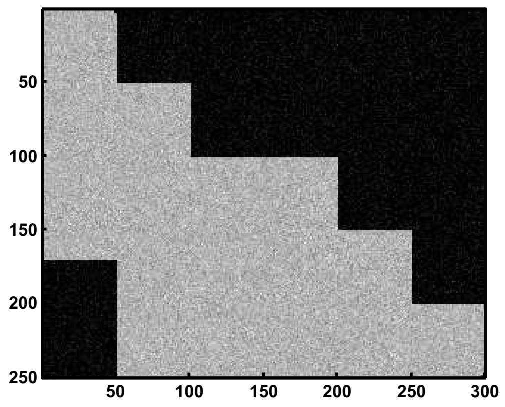
Example 4.2.3. This example treats the data first presented in Example 4.1.5 as rectangular data. Again, these data consist of 1,984 records of 16 key votes by the 435 members of the U.S. House of Representatives. I coded the data as 1 for yea, 0 for nay, and 0.5 for unknown. Figure 4.17(a) shows the raw rectangular data for this



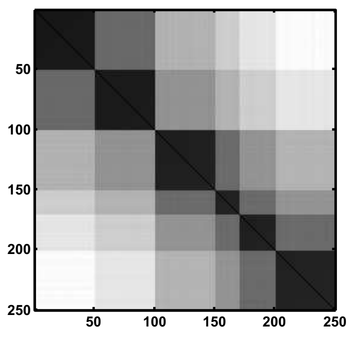
(a) Dissimilarity data



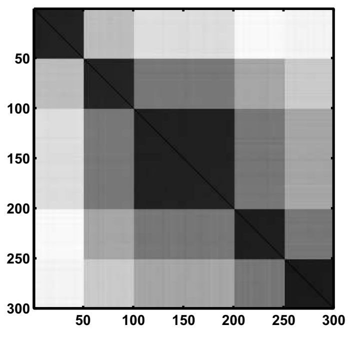
(b) $\mathbf{D}^*$



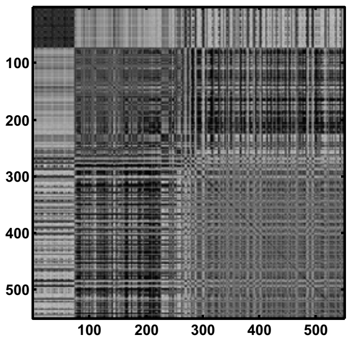
(c) alternate $\mathbf{D}^*$



(d) $\mathbf{D}^*_r$



(e) $\mathbf{D}^*_c$



(f) $\mathbf{D}^*_{r\cup c}$

Figure 4.16: Example 4.2.2 Pure rectangular relational data. (a) - dissimilarity data; (b) original co-VAT reordering, (c) alternate co-VAT reordering, (d-f) - co-VAT reordered dissimilarity data matrices.

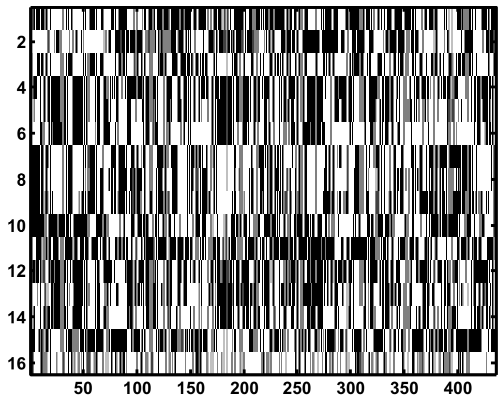
example.

Figures 4.17(b,c) show the co-VAT and alternate co-VAT reordered matrices, respectively. Again, the alternate co-VAT reordering produces a more appealing image, from the standpoint of assessing cluster tendency. There are clearly two larger co-clusters — one at the upper-left and on at the middle-bottom of view (c). These two co-clusters correspond to Republicans and Democrats, respectively, that tend to vote along party lines as well as the votes on which they tend to agree on. View (b) clearly does not provide as pleasing a result as view (c) does. Figure 4.18 shows the labels of the column objects, plotted in the order of column objects in view (b) and (c) of Fig. 4.17, respectively. The plots in Fig. 4.18 clearly support the visual evidence that the alternate co-VAT reordering is superior to the original formulation. As the figures show, the alternate algorithm is able to show how the members of each party cluster together; the original co-VAT ordering shows no such pattern.

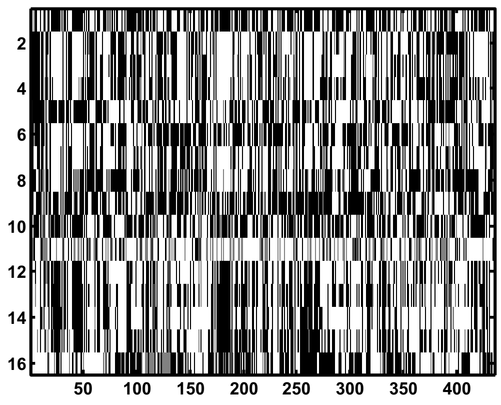
Figures 4.17(d-f) show the other co-VAT images. Note that view (e) is the same dissimilarity data that was used in the CLODD example shown in Fig. 4.12. Again, this view shows the two clusters of party members that tend to vote along party lines (Figure 4.18(b) shows the labels (Republican and Democrat) according the reordering of $\mathbf{D}_c$). View (f) of Fig. 4.17 shows the cluster tendency in the union of the members and votes. This view does not seem to suggest any cluster structure in the union of the objects. The failure of VAT to elucidate the cluster structure from $\mathbf{D}_{r\cup c}$ is the reason that the co-VAT image in view (b) is inferior to the alternate co-VAT image shown in view (c).

4.2.2 co-iVAT

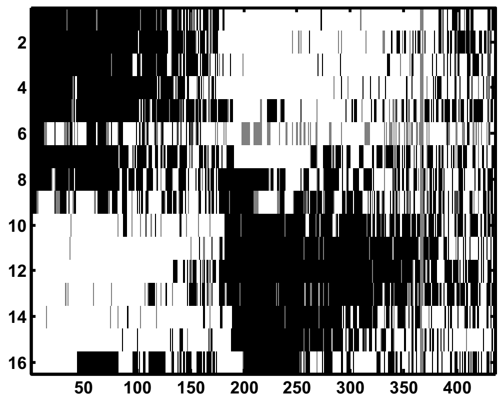
The iVAT distance transform in (2.4) is first applied to the three square co-VAT matrices, $\mathbf{D}_r^*$, $\mathbf{D}_c^*$, and $\mathbf{D}_{r\cup c}^*$, using the recursive formulation in Algorithm 3.4.1. The



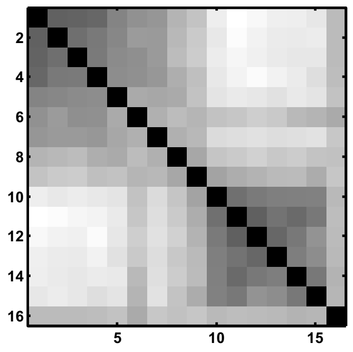
(a) Rectangular data



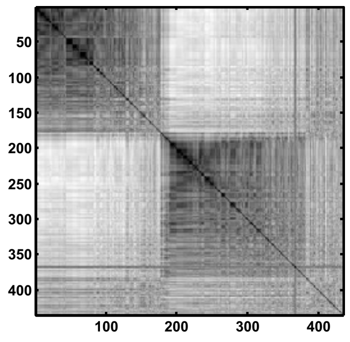
(b) $\mathbf{D}^*$



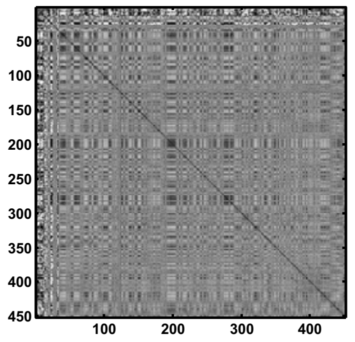
(c) alternate $\mathbf{D}^*$



(d) $\mathbf{D}_r^*$

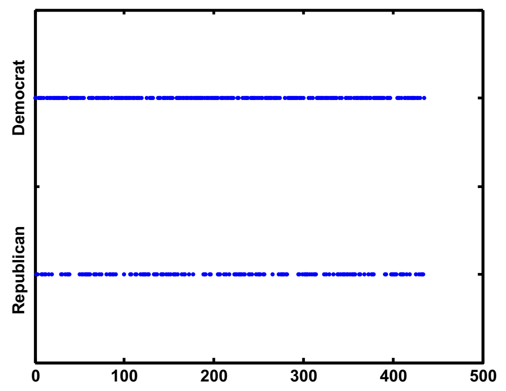


(e) $\mathbf{D}_c^*$

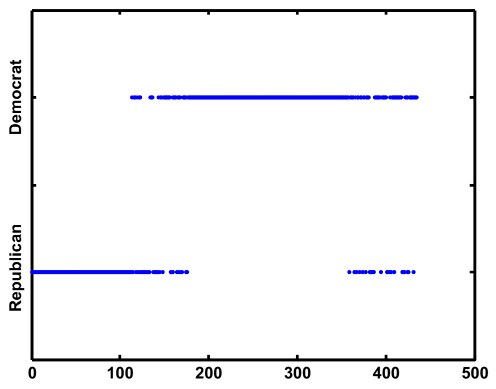


(f) $\mathbf{D}_{r\cup c}^*$

Figure 4.17: Example 4.2.3. (a) - rectangular data; (b) original co-VAT reordering, (c) alternate co-VAT reordering, (d-f) - co-VAT reordered dissimilarity data matrices.



(a) co-VAT reordered column object labels



(b) alternate co-VAT reordered column object labels

Figure 4.18: Labels ordered according to column object reordering of co-VAT and alternate co-VAT — shows that the alternate co-VAT discovers party affiliation pattern through voting records.

transformed matrices are denoted as $\mathbf{D}_r'^*$, $\mathbf{D}_c'^*$, and $\mathbf{D}_{r\cup c}'^*$ (examples of these matrices are shown in Figs. 4.20(a,c,d), respectively). Although, by definition, Eq. (2.4) could be applied to the rectangular dissimilarity matrix $\mathbf{D}$ by considering $\mathbf{D}$ to represent a partially-connected graph (edges only exist between row objects and column objects), applying this transform directly is computationally expensive. However, if one considers the elements of $\mathbf{D}_{r\cup c}'^*$ that correspond to elements of the rectangular matrix, one can build the reordered rectangular matrix $\mathbf{D}'^*$ from $\mathbf{D}_{r\cup c}'^*$.

The rectangular co-iVAT image is created as follows:

1. Build $\mathbf{D}_{r\cup c}$ and run VAT to produce $\mathbf{D}_{r\cup c}^*$, where the reordering indexes are $P_{r\cup c} = \{P(1), \ldots, P(m+n)\}$.

2. Compute $\mathbf{D}_{r\cup c}'^*$ from $\mathbf{D}_{r\cup c}^*$ using the recursive iVAT distance transform outlined in Algorithm 3.4.1.

3. Build the rectangular co-iVAT image $\mathbf{D}'$ from the corresponding elements of $\mathbf{D}_{r\cup c}'^*$. First, create the reordering indexes K and L, where K are the indexes of the elements of $P_{r\cup c} \leq m$ and L are the indexes of the elements of $P_{r\cup c} > m$

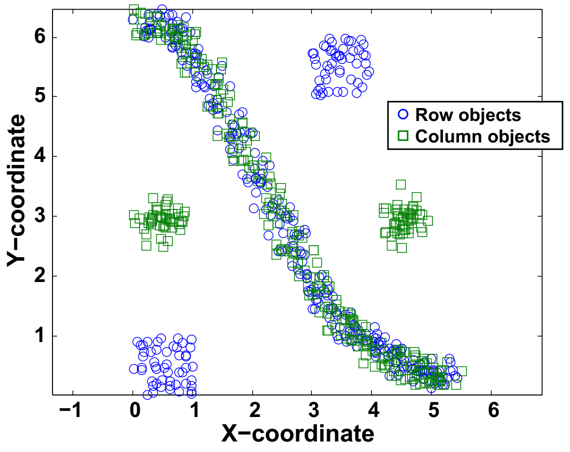
(m is the number of row objects). Then create $\mathbf{D}'^*$ by

$$\mathbf{D}'^* = \left[D'^*_{ij}\right] = \left[(D'^*_{r\cup c})_{K(i),L(j)}\right], 1 \leq i \leq m, 1 \leq j \leq n. \tag{4.10}$$

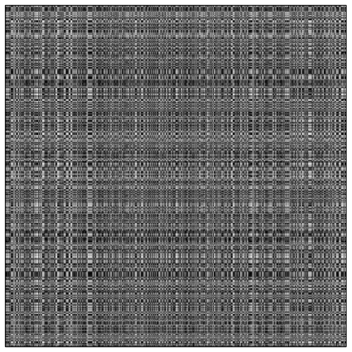
I have also adapted iVAT to the new reordering scheme of co-VAT presented in Algorithm 4.2.1. This adaptation requires the construction of $\mathbf{D}'^*_{r\cup c}$, as above, and the corresponding elements of $\mathbf{D}'^*_{r\cup c}$ are extracted to create the reordered rectangular matrix $\mathbf{D}'^*$. The co-VAT matrices built with the iVAT distance transform are called co-iVAT images.

Example 4.2.4. This numerical example is composed of 360 row objects and 360 column objects, as displayed in Fig. 4.19(a). The associated dissimilarity data, calculated using Euclidean distance, is shown in Fig. 4.19(b). The column objects (shown as green squares) are composed of three groups, the two groups of 50 objects located around coordinates (1.5,3) and (4.5,3), and the 260 objects organized along the curved line extending from the upper-left to the lower-right. The row objects (shown as blue circles) are composed of three groups, the two groups of 50 objects located around coordinates (1.5,0.5) and (3.5,5.5), and the 260 objects organized along the curved line extending from the upper-left to the lower-right. Hence, this example has a preferable cluster tendency of 3 clusters of row objects, 3 clusters composed of column objects, 5 clusters in the union of the row and column objects, and 1 co-cluster.

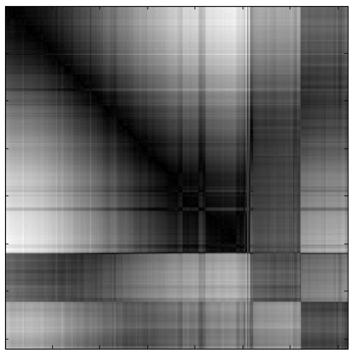
Figure 4.19(c,d) shows that both co-VAT and the alternate co-VAT display the 1 co-cluster as a diagonal band in the upper-left of the image, with both giving approximately equally pleasing results. The co-VAT images of $\mathbf{D}^*_r$ and $\mathbf{D}^*_c$ in Figs. 4.19(e,f), respectively, clearly show the smaller 2 clusters in each of the row objects and column objects as 2 smaller dark blocks in the lower-right of the image. Again the large co-cluster is shown as a dark diagonal band in the upper-left. The image of $D^*_{r\cup c}$ is



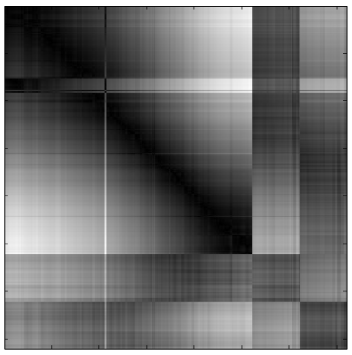
(a) Object Data



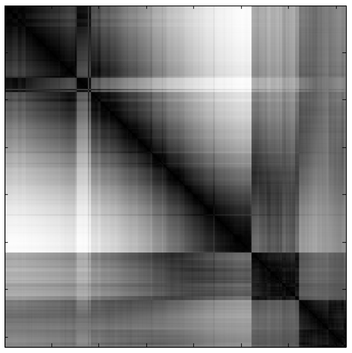
(b) Dissimilarity Data - $\mathbf{D}$



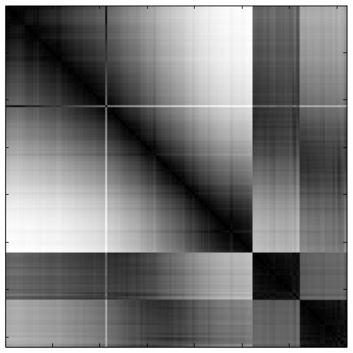
(c) co-VAT image - $\mathbf{D}^*$



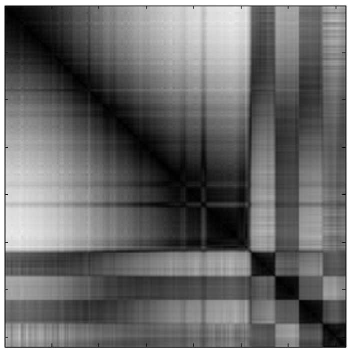
(d) Alternate-reordering co-VAT image - $\mathbf{D}^*$



(e) co-VAT image - $\mathbf{D}_r^*$

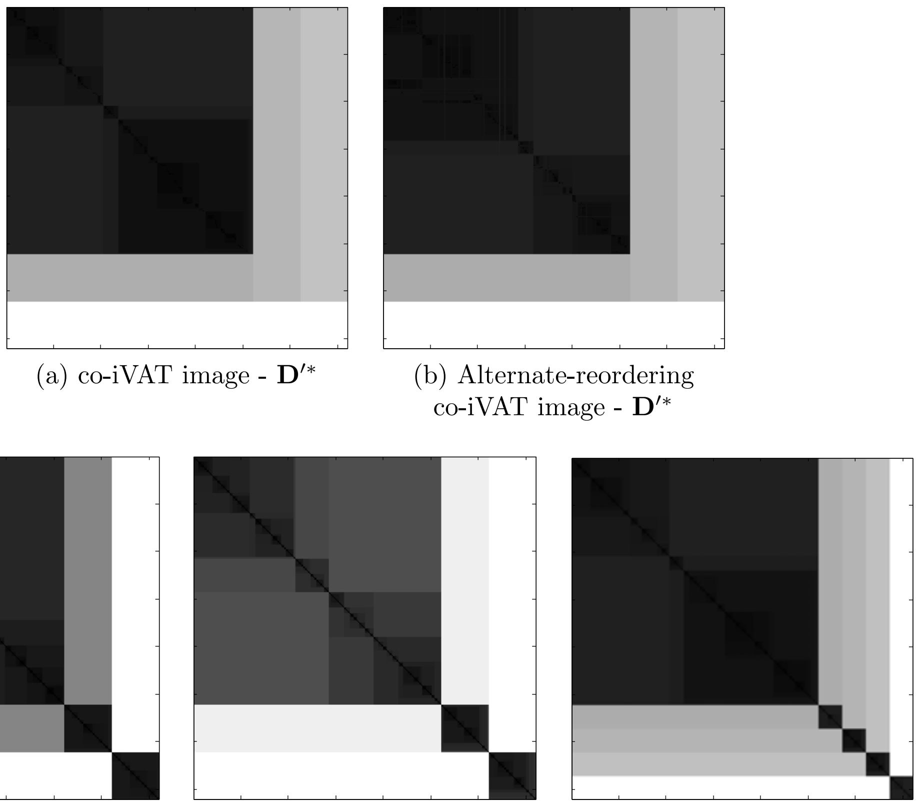


(f) co-VAT image - $\mathbf{D}_c^*$



(g) co-VAT image - $\mathbf{D}_{r\cup c}^*$

Figure 4.19: co-VAT images of 360 row objects and 360 column objects represented by rectangular dissimilarity data $\mathbf{D}$



(a) co-iVAT image - $\mathbf{D}'^{*}$

(b) Alternate-reordering co-iVAT image - $\mathbf{D}'^{*}$

(c) co-iVAT image - $\mathbf{D}_r'^{*}$

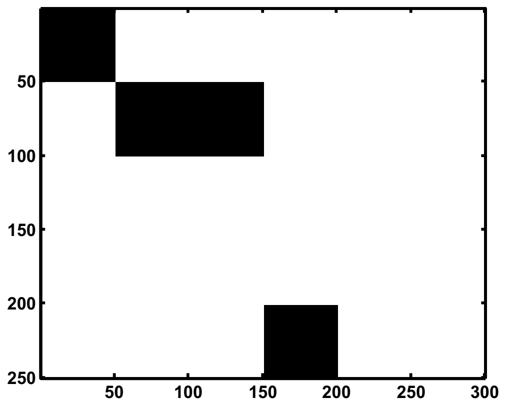
(d) co-iVAT image - $\mathbf{D}_c'^{*}$

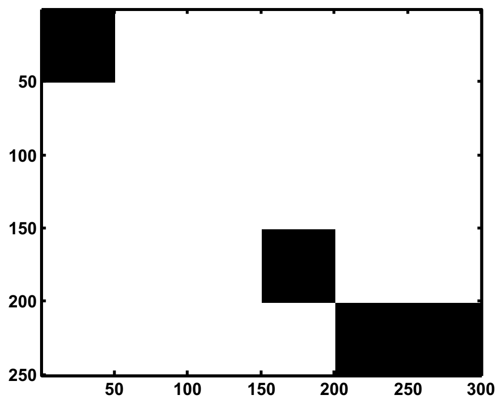
(e) co-iVAT image - $\mathbf{D}_{r\cup c}'^{*}$

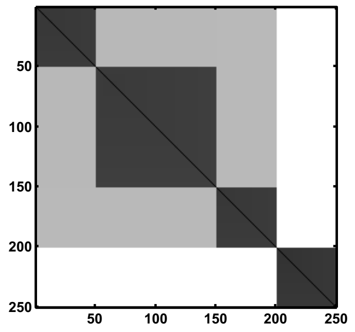
Figure 4.20: co-iVAT images of 360 row objects and 360 column objects represented by rectangular dissimilarity data $\mathbf{D}$

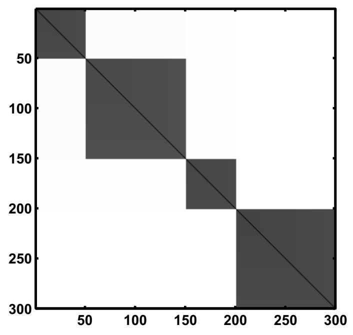
shown in Fig. 4.19(g); it shows the 5 clusters in $O_r \cup O_c$ as the four dark blocks in the lower-right and the dark diagonal band in the upper-left. While we hesitate to say that co-VAT and the alternate co-VAT have failed for this example, we believe that the large diagonal band leads to ambiguity as to the cluster tendency of these data. Further more, the contrast in Figs. 4.19(c,d) make it difficult to determine the number of co-clusters.

Figure 4.20 shows the corresponding co-iVAT images of the rectangular dissimilarity data shown in Fig. 4.19. The co-iVAT images give a very clear view of the cluster tendency for each of the four types of clusters: $\mathbf{D}'^{*}$ shows 1 co-cluster, $\mathbf{D}_r'^{*}$ shows 3 row-clusters, $\mathbf{D}_c'^{*}$ shows 3 column-clusters, and $\mathbf{D}_{r\cup c}'^{*}$ shows 5 clusters in $O_r \cup O_c$.


(a) $\mathbf{D}'^{*}$


(b) alternate $\mathbf{D}'^{*}$


(c) $\mathbf{D}'^{*}_{r}$


(d) $\mathbf{D}'^{*}_{c}$

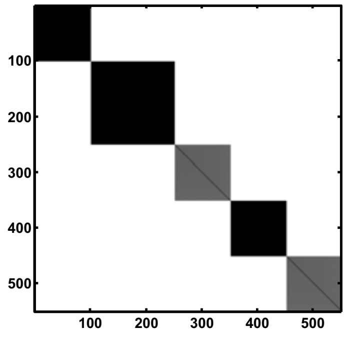
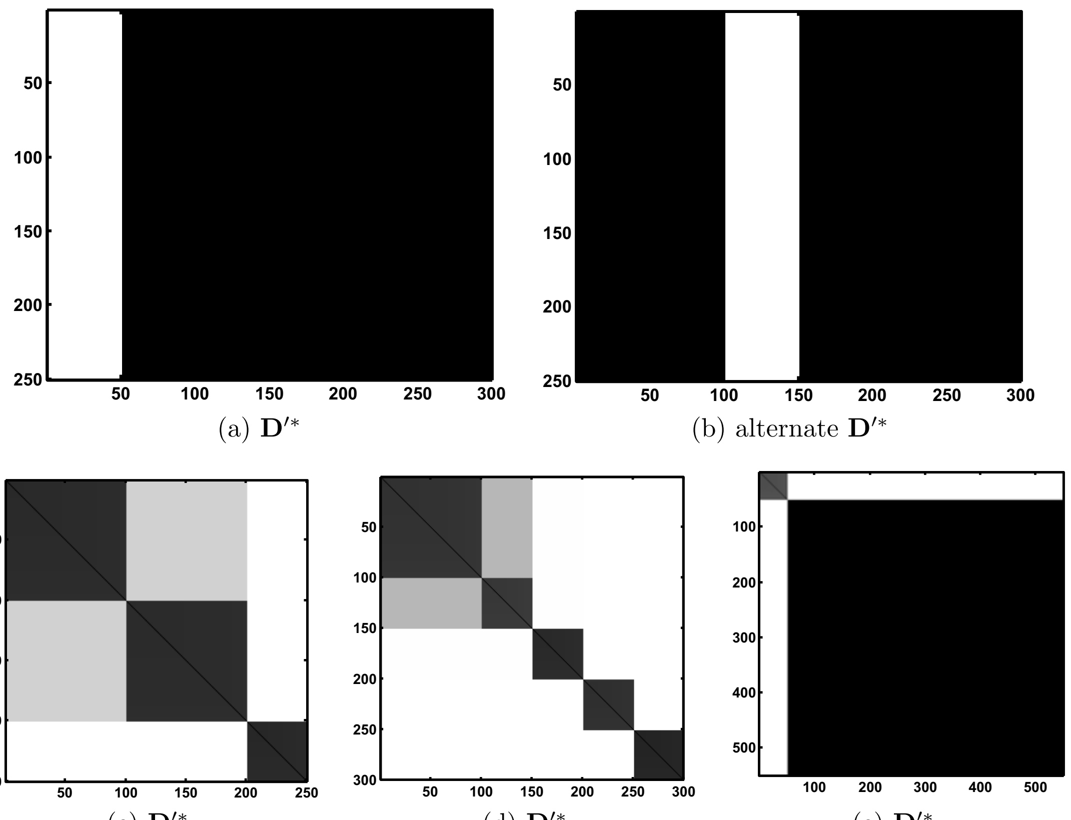

(e) $\mathbf{D}'^{*}_{r\cup c}$

Figure 4.21: co-iVAT images of dissimilarity data in Fig. 2.10; (a) original co-iVAT reordering, (b) alternate co-iVAT reordering, (c-e) co-iVAT reordered dissimilarity data matrices.

Figures 4.21, 4.22, and 4.23 illustrate the co-iVAT images for the pure relational data examples first shown in Fig. 2.10 and Examples 4.2.1 and 4.2.2. These examples highlight my conjecture that there are actually only 1 pure co-cluster in Examples 4.2.1 and 4.2.2.

The co-iVAT images in Fig. 4.21 are very similar to results achieved using using the conventional co-VAT method. However, the contrast of these images is improved; although, the improvement is slight. There is a drastic difference between the co-iVAT images in Figs. 4.22 and 4.23 to the co-VAT images in Figs. 4.15 and 4.16.

First, the co-iVAT images shown in Fig. 4.22 clearly suggest that there is 1 pure co-cluster in the data. I would like to point out that in this case the original co-VAT

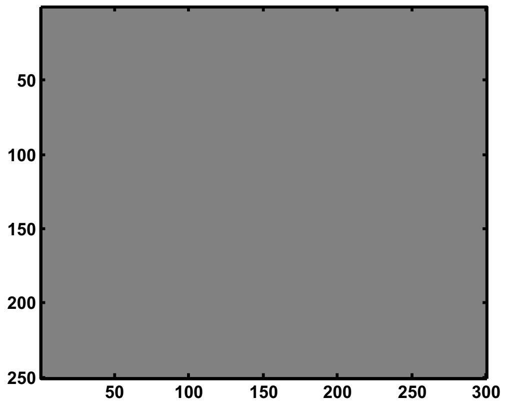

(a) $\mathbf{D}'^*$ (b) alternate $\mathbf{D}'^*$

(c) $\mathbf{D}_r'^*$ (d) $\mathbf{D}_c'^*$ (e) $\mathbf{D}_{r\cup c}'^*$

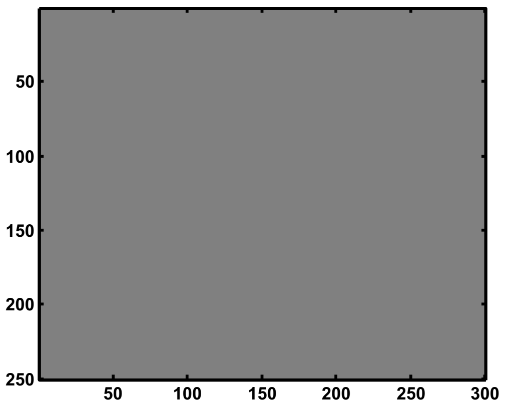
Figure 4.22: co-iVAT images of dissimilarity data in Example 4.2.1; (a) original co-iVAT reordering, (b) alternate co-iVAT reordering, (c-e) co-iVAT reordered dissimilarity data matrices.

ordering scheme, in view (a), is more pleasing than the alternate reordering method, shown in view (b), because the large co-cluster is shown as a contiguous dark block rather than the broken dark block shown in view (b). Interestingly, this example contradicts the axiom that $c_p = c_{r\cup c} - c_{co} = c_r + c_c - c_{co}$. View (c) suggest $c_r = 3$, view (d) suggests $c_c = 5$, and view (e) suggests $c_{r\cup c} = 2$. Thus, $c_p = 3 + 5 - 2 = 6$. The tendency of $c_p = 6$ is clearly not suggested by views (a) and (b). I will address this question further in Section 4.3.

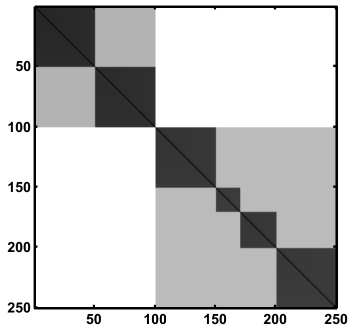
A similar conundrum exists in the example shown in Fig. 4.23. Views (a) and (b) show that *all* objects are grouped in one pure co-cluster. View (c) shows $c_r = 6$, view (d) shows $c_c = 5$, and view (e) shows that there is 1 cluster in the union of the row



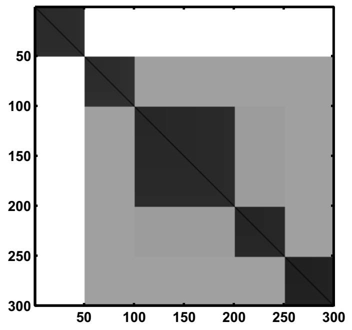
(a) $\mathbf{D}'^*$



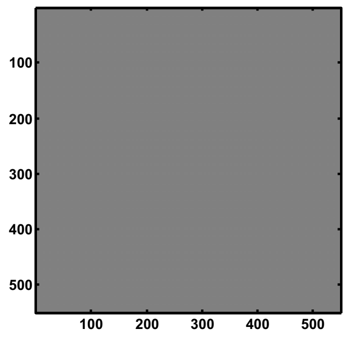
(b) alternate $\mathbf{D}'^*$



(c) $\mathbf{D}_r'^*$



(d) $\mathbf{D}_c'^*$

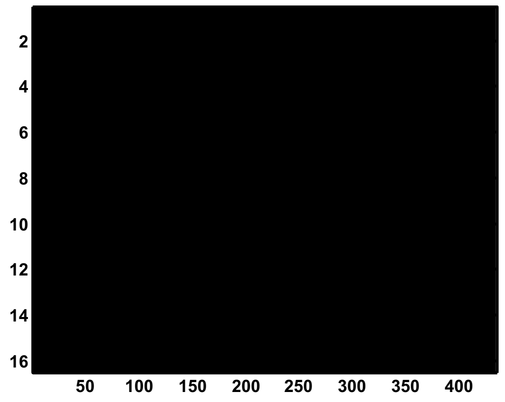


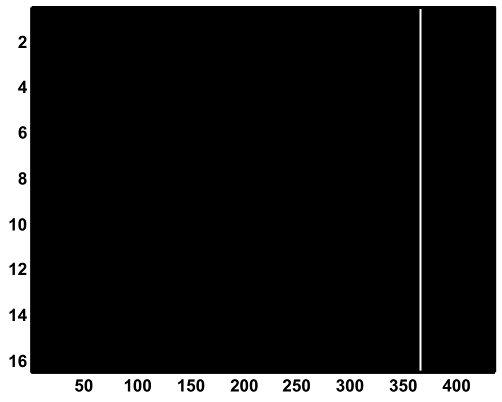
(e) $\mathbf{D}_{r\cup c}'^*$

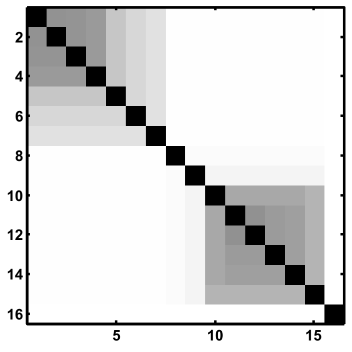
Figure 4.23: co-iVAT images of dissimilarity data in Example 4.2.2; (a) original co-iVAT reordering, (b) alternate co-iVAT reordering, (c-e) co-iVAT reordered dissimilarity data matrices.

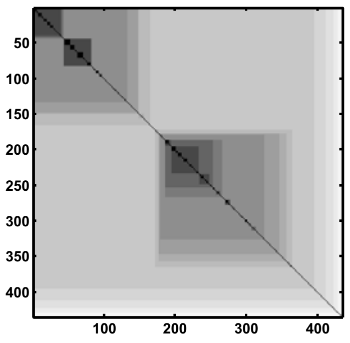
and column objects. Thus $c_p = 6 + 5 - 1 = 10$. Obviously, views (a) and (b) do not suggest $c_p = 10$. I will address this discrepancy further in the next section.

Example 4.2.5. Figure 4.24 shows the co-iVAT images of the dissimilarity data first presented in Example 4.2.3. Views (a) and (b) show the co-iVAT images of the reordered dissimilarity data. It is my opinion that these images are an example where co-iVAT fails to show the cluster structure in $\mathbf{D}$. This is in contrast to the co-VAT image shown in Fig. 4.17(c), which more clearly shows the grouping of the row and column objects (key-votes and representatives, in this case). The co-iVAT images shown in views (c) and (d) show the cluster tendency in $\mathbf{D}_r$ (votes) and $\mathbf{D}_c$


(a) $\mathbf{D}'^{*}$


(b) alternate $\mathbf{D}'^{*}$


(c) $\mathbf{D}'^{*}_{r}$


(d) $\mathbf{D}'^{*}_{c}$

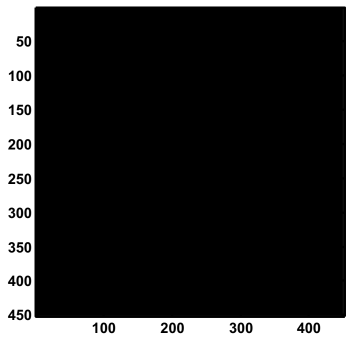

(e) $\mathbf{D}'^{*}_{r\cup c}$

Figure 4.24: co-iVAT images of VOTE rectangular data in Example 4.2.3; (a) original co-iVAT reordering, (b) alternate co-iVAT reordering, (c-e) co-iVAT reordered dissimilarity data matrices.

(representatives), respectively. These images show similar structure as shown by the respective co-VAT images shown in Figs.4.17(d,e) — 2 clusters in the row objects and 2 clusters in the column objects. Finally, view (e) shows the co-iVAT image of $\mathbf{D}_{r\cup c}$. This image suffers from the same problem as seen in the co-iVAT images shown in views (a) and (b); namely, the cluster structure is not shown due to the distance transform. This problem occurs because of the tertiary nature of this data set (yea, nay, and unknown). If two representatives have at least one 'nay' vote in common, then the path-based distance is 0. This is the issue with using the path-based distance with this data-set.

The last example, the VOTE data set, showed a case where the iVAT distance transform degraded the effectiveness of co-VAT in showing the cluster structure. However, as I proved in Section 3.4, the computational complexity of VAT and iVAT are equivalent. Note that proofs in Section 3.4 can also be extended to show that the co-iVAT images are in the same order as the co-VAT images; only the dissimilarity values themselves are altered. Thus, I recommend computing both co-VAT and co-iVAT images of a rectangular data set and using both sets of images to judge the cluster structure.

4.3 Rectangular Single-Linkage

Rectangular Single-Linkage (ReSL) is similar to CLODD, in that ReSL partitions ordered dissimilarity data. However, unlike CLODD, ReSL addresses rectangular dissimilarity data. The front end ordering algorithm for ReSL is the alternate co-VAT, presented in Section 4.2. The alternate co-VAT reorders the dissimilarity data and displays cluster tendency among row objects $\mathbf{O}_r$, column objects $\mathbf{O}_c$, the union of row and column objects $\mathbf{O}_r \cup \mathbf{O}_c$, and the co-clusters. Again, co-clusters are defined as clusters among the union of the row and column objects that contain at least one object of each type.

The name *Rectangular Single-Linkage* comes from the fact that CLODD is used to extract aligned partitions from the co-VAT image. It was shown in Chapter 3 that, in fact, these aligned partitions are SL clusters—[CLODD + VAT] is essentially [SL + a heuristic for finding the best SL partition]. Thus, I extend this line of thought to rectangular data; thus, CLODD for rectangular data is ReSL.

Consider $n = n_r + n_c$ objects, $\mathbf{O} = \mathbf{O}_r \cup \mathbf{O}_c$. These objects are represented by an $n_r \times n_c$ rectangular dissimilarity matrix $\mathbf{D} \in [0, 1]^{n_r n_c}$ (assume that $\mathbf{D}$ has been normalized to the interval $[0,1]$). ReSL begins by computing the co-VAT matrices $\mathbf{D}^*_r$, $\mathbf{D}^*_c$, $\mathbf{D}^*_{r\cup c}$, and $\mathbf{D}^*$. Note that the alternate co-VAT reordering method described in Section 4.2 is required for ReSL as the alternate formulation maintains consistent ordering between $\mathbf{D}^*$, $\mathbf{D}^*_r$, and $\mathbf{D}^*_c$. Then, I use CLODD to compute aligned partitions of the three square relational matrices, $\mathbf{D}^*_r$, $\mathbf{D}^*_c$, and $\mathbf{D}^*_{r\cup c}$. These partitions are denoted as $\mathbf{U}^*_r = \{(n_r)_1 : \ldots : (n_r)_{k_r}\}$, $\mathbf{U}^*_c = \{(n_c)_1 : \ldots : (n_c)_{k_c}\}$, and $\mathbf{U}^*_{r\cup c} = \{(n_{r\cup c})_1 : \ldots : (n_{r\cup c})_{k_{r\cup c}}\}$, respectively.

The definition of a co-cluster is a cluster that is composed of both row objects and column objects. Thus, I use the partitions $\mathbf{U}^*_r$ and $\mathbf{U}^*_c$ to determine which of the clusters, when considered together, are composed of mixed row and column objects. If one considers the aligned partition of the row objects in respect to the rectangular data $\mathbf{D}^*$—when $\mathbf{D}^*$ is reordered with the alternate reordering method—the partition appears as horizontal boundaries, as shown by the yellow horizontal lines in Fig. 4.25(d). The aligned partition of the column objects in respect to $\mathbf{D}^*$ appear as vertical boundaries, as shown by the yellow vertical lines in Fig. 4.25(d). In the example shown in Fig. 4.25, there are 3 row clusters and 4 column clusters; thus, there are 12 combinations of the row and column clusters that could be co-clusters. Hence, I compute the degree of co-clusterness by first calculating the mean-value of the dissimilarity data within each candidate co-cluster,

$$(d_{co})_{ij} = \frac{1}{(n_r)_i (n_c)_j} \sum_{k:(U^*_r)_{ik}=1} \sum_{l:(U^*_c)_{jl}=1} D^*_{kl}. \tag{4.11}$$

This equation is the mean-value of the of the rectangular dissimilarity data within each possible co-cluster (combination of a row and column cluster) or, in other words, the mean value of the dissimilarity between a row and column cluster. Thus, if this value is small, then the degree of co-clusterness is large. Because normalized dissimilarity data is assumed, I calculate the degree of co-clusterness by the transformation

$$\mathbf{U}_{co} = 1 - d_{co}. \tag{4.12}$$

At this point in the algorithm, one can use the degree of co-clusterness matrix to represent the co-clusters (much as one would with a possibilistic partition matrix). However, an estimate of the number of co-clusters k_{co} can be obtained by $k_{co} = k_r + k_c - k_{r\cup c}$, where k_r, k_c, and $k_{r\cup c}$ are the number of clusters in each square co-VAT matrix as returned by CLODD. The co-clusters are the objects corresponding to the k_{co} greatest elements of $\mathbf{U}_{co}$.

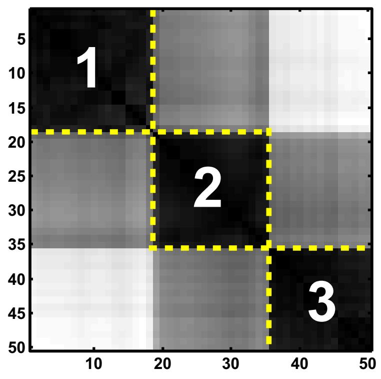
Algorithm 4.3.1 outlines the steps of ReSL.

Algorithm 4.3.1: Rectangular Single-Linkage

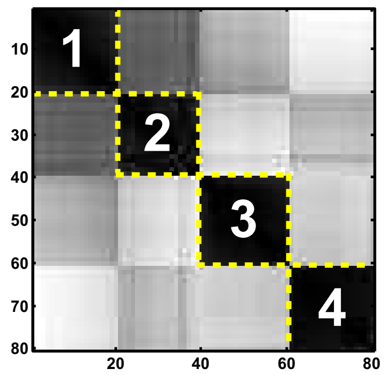
```
  Input: $\mathbf{D} \in \mathbb{N}^{n_r n_c}$ - rectangular dissimilarity data
  Data: Partition matrices - $\mathbf{U}_r^* \in \mathbf{M}_{hk_r n_r}$, $\mathbf{U}_c^* \in \mathbf{M}_{hk_c n_c}$, $\mathbf{U}_{r\cup c}^* \in \mathbf{M}_{hk_{r\cup c} n}$, and
        $\mathbf{U}_{co}$
1 Compute co-VAT matrices, $\mathbf{D}_r^*$, $\mathbf{D}_c^*$, $\mathbf{D}_{r\cup c}^*$, and $\mathbf{D}^*$.
2 Use CLODD to compute $\mathbf{U}_r^* = \{(n_r)_1 : \ldots : (n_r)_{k_r}\}$.
3 Use CLODD to compute $\mathbf{U}_c^* = \{(n_c)_1 : \ldots : (n_c)_{k_c}\}$.
4 Use CLODD to compute $\mathbf{U}_{r\cup c}^* = \{(n_{r\cup c})_1 : \ldots : (n_{r\cup c})_{k_{r\cup c}}\}$.
5 Compute $\mathbf{U}_{co}$ using (4.12)
  If the number of co-clusters is desired, $k_{co} = k_r + k_c - k_{r\cup c}$.
6 Co-clusters are objects corresponding to $k_{co}$ greatest elements of $\mathbf{U}_{co}$.
```

4.3.1 Numerical Examples

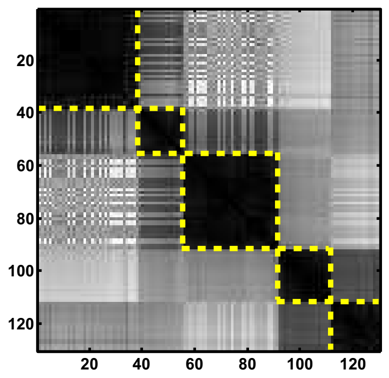
Example 4.3.1. This example demonstrates the ReSL algorithm on the rectangular data first shown in Fig. 2.9. The rectangular dissimilarity data $\mathbf{D}$ was built using Euclidean distance on 50 row objects and 80 column objects. The results of the ReSL algorithm are shown in Fig. 4.25. As view (e) shows, there are 3 clusters in the row objects (blue circles), 4 clusters in the column objects (green squares), 5 clusters in



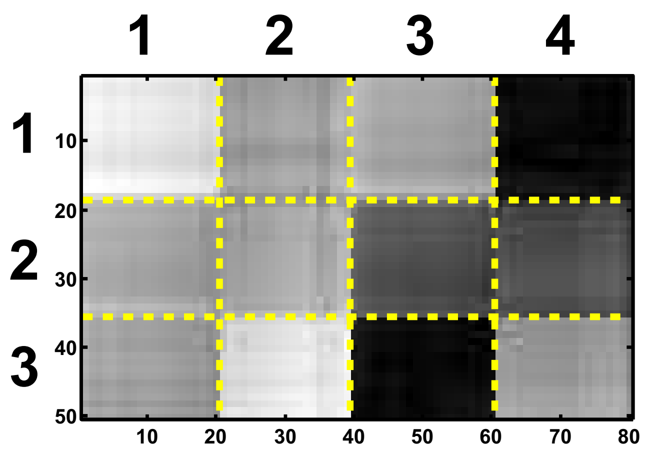
(a) Partition of $\mathbf{D}_r^*$



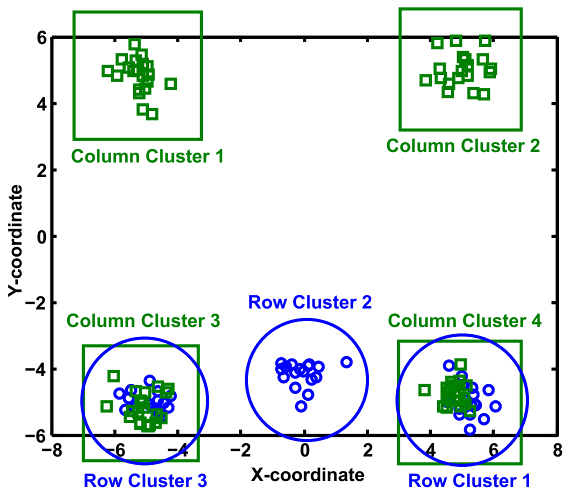
(b) Partition of $\mathbf{D}_c^*$



(c) Partition of $\mathbf{D}_{r\cup c}^*$



(d) Partition of $\mathbf{D}^*$



(e) Object data, ∘ - row objects, □ - column objects

$$\begin{bmatrix} 0 & 0.30 & 0.28 & 0.94 \\ 0.27 & 0.27 & 0.64 & 0.66 \\ 0.31 & 0.01 & 0.94 & 0.32 \end{bmatrix}$$

(f) Partition values of $\mathbf{U}_{co}$

Figure 4.25: ReSL partitions of co-VAT reordered dissimilarity images shown in Fig. 2.9

the union of the objects, and 2 co-clusters (the two groups at the lower-left and lower-right). The results of the ReSL algorithm support this visually preferred clustering of this data set.

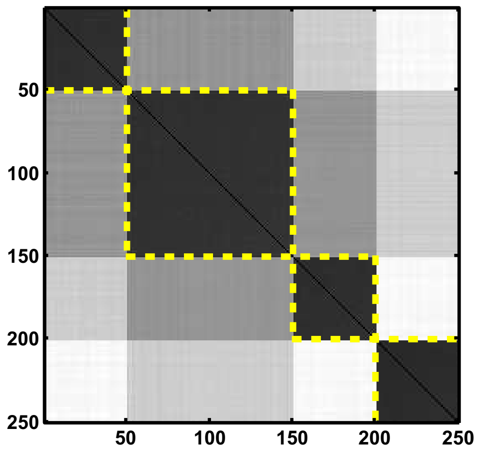
View (a) of Fig. 4.25 illustrates the 3-partition computed by CLODD from the co-VAT reordered dissimilarity matrix $\mathbf{D}_r^*$. This partition is also shown by the three larger blue circles in the lower part of the plot in view (e). View (b) shows that CLODD is able to find the preferred 4-partition of the column clusters, which are also shown as the four larger green squares in view (e). The numbers shown in views (a) and (b) correspond to the number of the row and column clusters in view (e). There are 5 clusters in the union of the objects, and view (c) shows that CLODD is able to find this preferred partition in the co-VAT reordered dissimilarity matrix $\mathbf{D}_{r\cup c}^*$. Finally, as described in Algorithm 4.3.1, the partitions of the row objects and column objects are used to find the degrees of co-clusterness for each group of row and column objects. View (d) shows the candidate co-cluster boundaries in $\mathbf{D}^*$ found by the ReSL algorithm. Clearly, ReSL is able to find the visually-evident boundaries in $\mathbf{D}^*$. Finally, these boundaries are used to compute the degree of co-clusterness for each region. The degree of co-clusterness matrix $\mathbf{U}_{co}$ is shown in view (f). The matrix $\mathbf{U}_{co}$ shows that the two co-clusters, the groups at the lower-left and lower-right in view (e), have the highest degree of co-clusterness at 0.94. The combinations of the center row cluster with the two column clusters at the lower-right and lower-left produce the next highest degrees of co-clusterness, which is expected. If one chooses to "harden" $\mathbf{U}_{co}$ by choosing the two $(k_r + k_c - k_{r\cup c})$ highest degrees, then ReSL is clearly able to find the two co-clusters in this example.

Example 4.3.2. This example, shown in Fig. 4.26, demonstrates ReSL on the data first presented in Example 2.3.2. Views (a) and (b) show the partitions of the row objects and the column objects, respectively. CLODD is clearly able to find the 4 dark blocks on the diagonal of these images. The 4 clusters in the row objects are the row-indexes $\{1-50\}, \{51-150\}, \{151-200\}, \{201-250\}$ in view (d). The 4 clusters in the column objects are the column-indexes $\{1-50\}, \{51-150\}, \{151-200\}, \{201-250\}$ in view (d). It is interesting that the row objects and column objects indexed $\{51-250\}$ appear as a cluster in $\mathbf{D}_r^*$ and $\mathbf{D}_c^*$. But this is expected, because the clustering of $\mathbf{D}_r^*$ and D_c^* can be thought of clustering the row-vectors and column-vectors in $\mathbf{D}^*$, where these vectors are the dissimilarity values in the rows and columns of this matrix.

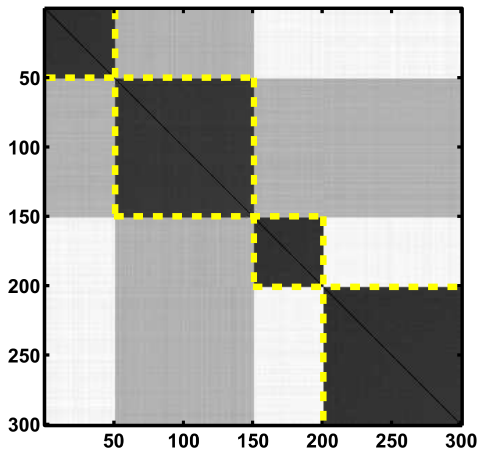
View (c) shows the partition of the union of the row and column objects. This image shows that CLODD returns the visually appealing 5-partition of $\mathbf{D}_{r\cup c}^*$. According to the rule that the number of co-clusters $k_{co} = k_r + k_c - k_{r\cup c}$, the number of co-clusters should be $3 = 4 + 4 - 5$. The degree of co-clusterness matrix, shown in view (e), supports the claim of 3 co-clusters by the 3 large values (0.96).

Example 4.3.3. Figure 4.27 illustrates the ReSL partitions of the rectangular data from Example 4.2.1. This example is interesting because, as was stated previously, it is my opinion that this data set contains 1 co-cluster. This was supported by the co-iVAT image shown in Figs. 4.22(a,b). ReSL finds 3 clusters in the image of $\mathbf{D}_r^*$ in view (a) and 5 clusters in the image of $\mathbf{D}_c^*$ in view (b), which, in both cases, is the visually preferable solution. However, the image of $\mathbf{D}_{r\cup c}^*$ does not show any clear cluster structure, expect for perhaps the one distinct dark block in the upper-left. Like any clustering algorithm will, CLODD returned a partition for this dissimilarity matrix — view (c) shows that CLODD returned a 3-partition: one very large cluster, and two smaller clusters. Thus, according to the co-cluster rule, the number of co-clusters is $k_{co} = 3 + 5 - 3 = 5$. However, the degree of co-clusterness matrix in view (e) shows that there are 6 dark blocks in $\mathbf{D}^*$, as illustrated in view (d). Does this mean that there are actually 6 co-clusters in this data set?

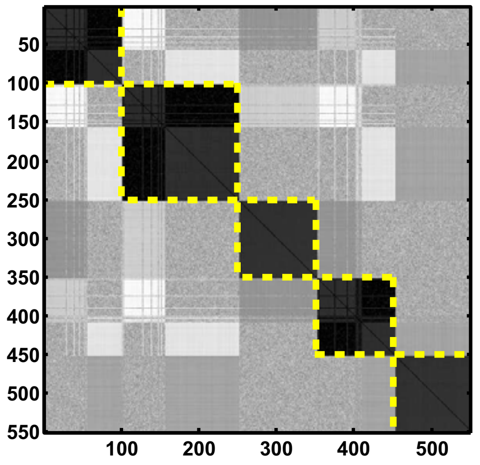
It is still my opinion that there is 1 co-cluster in this data, because the 6 co-clusters



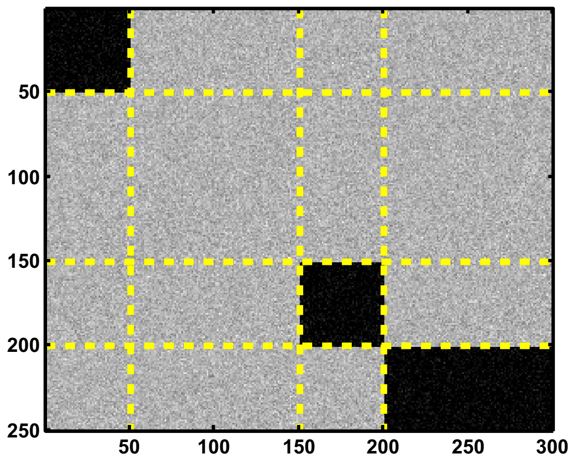
(a) Partition of $\mathbf{D}_r^*$



(b) Partition of $\mathbf{D}_c^*$



(c) Partition of $\mathbf{D}_{r\cup c}^*$

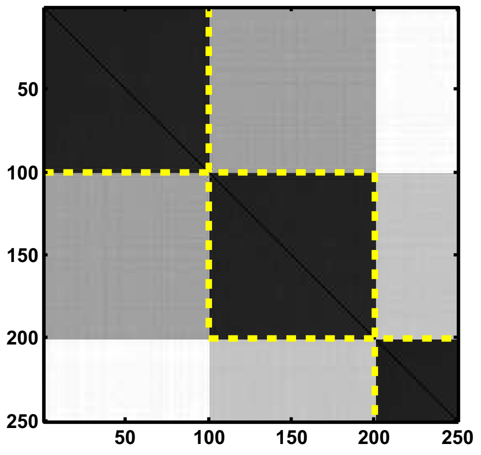


(d) Partition of $\mathbf{D}^*$

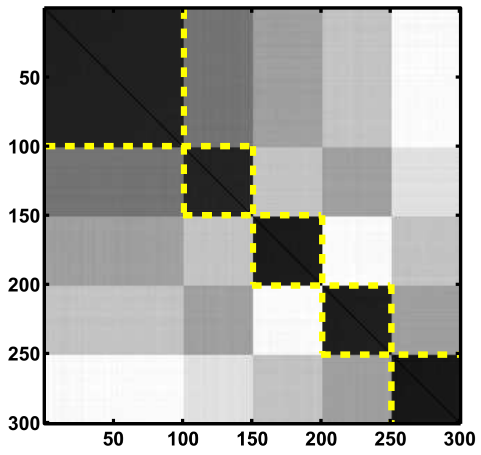
$$\begin{bmatrix} 0.96 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.96 & 0 \\ 0 & 0 & 0 & 0.96 \end{bmatrix}$$

(e) Partition values of $\mathbf{U}_{co}$

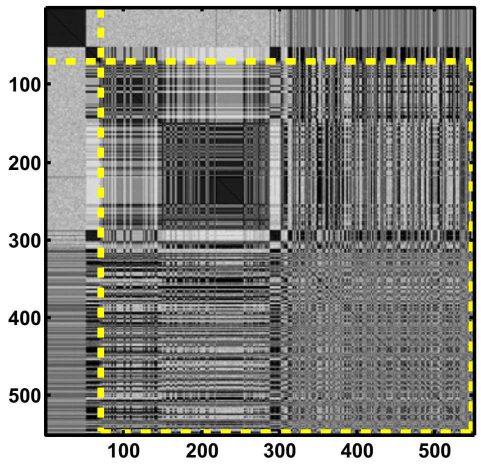
Figure 4.26: ReSL partitions of co-VAT reordered dissimilarity images shown in Fig. 2.10



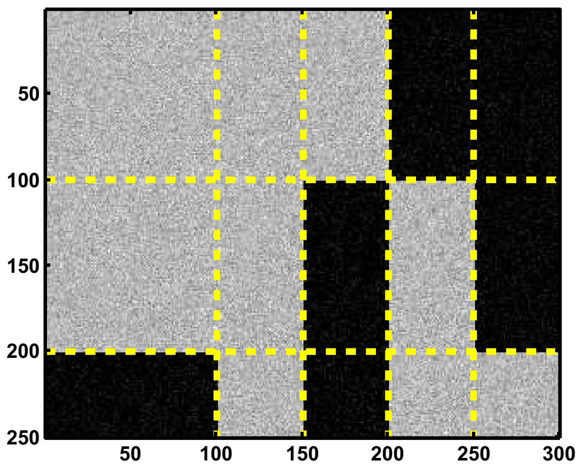
(a) Partition of $\mathbf{D}_r^*$



(b) Partition of $\mathbf{D}_c^*$



(c) Partition of $\mathbf{D}_{r\cup c}^*$



(d) Partition of $\mathbf{D}^*$

$$\begin{bmatrix} 0 & 0 & 0 & 0.96 & 0.96 \\ 0 & 0 & 0.96 & 0 & 0.96 \\ 0.96 & 0 & 0.96 & 0 & 0 \end{bmatrix}$$

(e) Partition values of $\mathbf{U}_{co}$

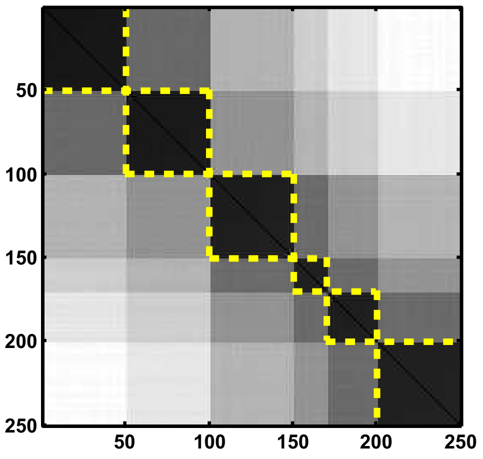
Figure 4.27: ReSL partitions of co-VAT reordered dissimilarity images shown in Fig. 4.15

found by ReSL are all overlapped — the row objects indexed $\{1 - 100\}$ are similar in relation to the column objects indexed $\{250 - 300\}$, these column objects are also similar to the row objects $\{100 - 200\}$, and so on.

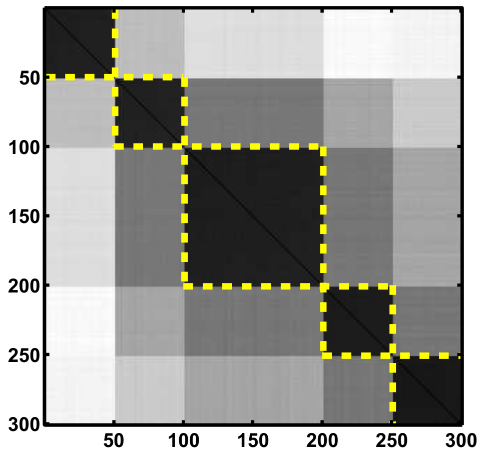
Example 4.3.4. Figure 4.28 shows the ReSL partitions of the rectangular data first presented in Example 4.2.2. This example is very similar to the previous example, as there is, in my opinion, 1 co-cluster which ReSL detects as many overlapping co-clusters. Views (a) and (b) show that CLODD is able to partition the row and column objects, respectively, in the visually preferred manner — there are 6 clusters in the row objects and 5 clusters in the column objects. If one thinks of the rows of $\mathbf{D}^*$ as vectors, then view (d) shows that there are indeed 6 distinct patterns in the rows. View (d) also supports the claim of 5 clusters in the column objects. Again, as was seen in the previous example, the co-VAT image of $D^*_{r\cup c}$ does not show any particular cluster structure — CLODD finds 3 clusters in $\mathbf{O}_r \cup \mathbf{O}_c$. Thus, the rule-of-thumb suggests that there are $8 = 6 + 5 - 3$ co-clusters. However, the degree of co-clusterness matrix show in view (e) does not support this claim, and instead suggests that there are 13 co-clusters. Although, again, it is my opinion that there is 1 co-cluster, which in this case encompasses all 550 objects.

Example 4.3.5. This last example presents the ReSL partitions of the VOTE rectangular data set, first presented in Example 4.2.3. Figure 4.29 shows the results of ReSL on the VOTE data. The CLODD partitions of $\mathbf{D}^*_r$ and $\mathbf{D}^*_c$, in views (a) and (b), are interesting. The votes (row objects) are partitioned into 10 clusters, with 8 clusters containing only one object. The partition of $\mathbf{D}^*_c$ returns 3 clusters. The two large dark blocks in $\mathbf{D}^*_c$ represent the two groups of representatives that vote along party lines and the group that votes independently.

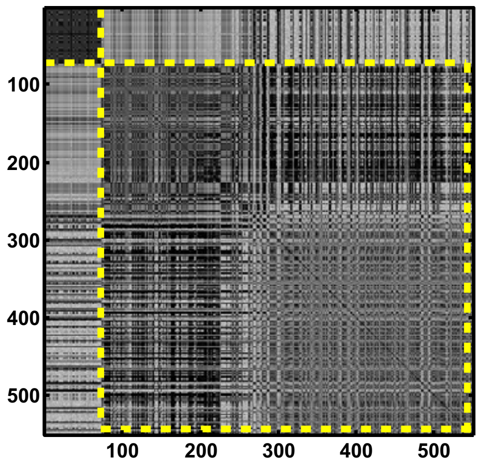
The partitions in views (a) and (b) are projected into the rectangular data, as outlined in the ReSL algorithm descrition. The co-cluster partitions are shown in



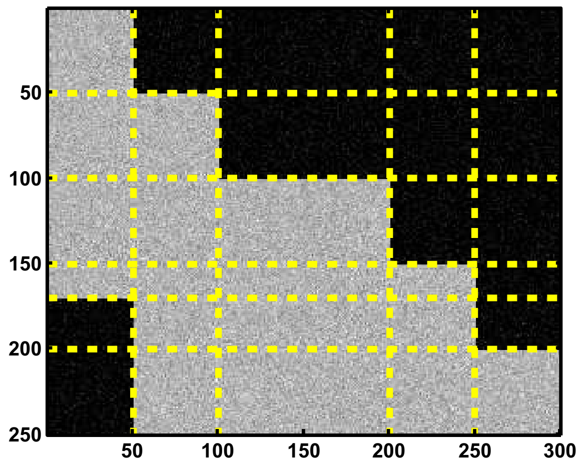
(a) Partition of $\mathbf{D}_r^*$



(b) Partition of $\mathbf{D}_c^*$



(c) Partition of $\mathbf{D}_{r\cup c}^*$

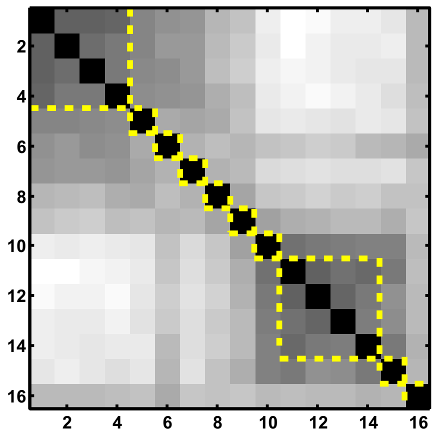


(d) Partition of $\mathbf{D}^*$

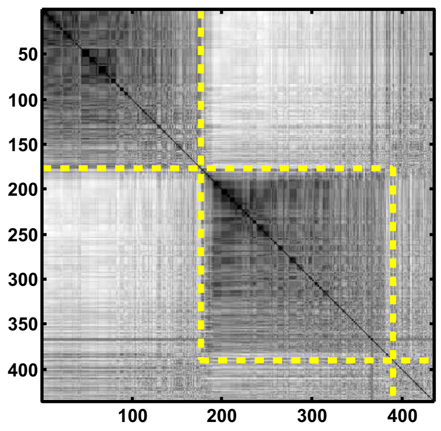
$$\begin{bmatrix} 0 & 0.96 & 0.96 & 0.96 & 0.96 \\ 0 & 0 & 0.96 & 0.96 & 0.96 \\ 0 & 0 & 0 & 0.96 & 0.96 \\ 0 & 0 & 0 & 0 & 0.96 \\ 0.96 & 0 & 0 & 0 & 0.96 \\ 0.96 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(e) Partition values of $\mathbf{U}_{co}$

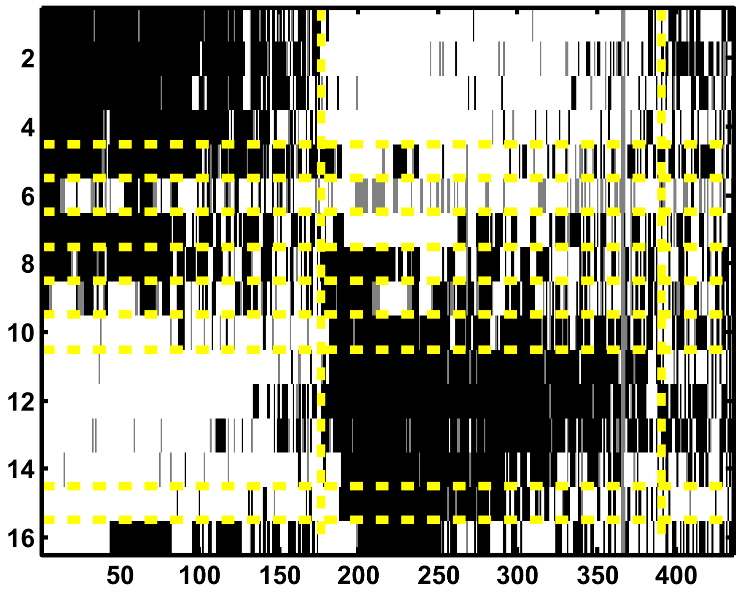
Figure 4.28: ReSL partitions of co-VAT reordered dissimilarity images shown in Fig. 4.16



(a) Partition of $\mathbf{D}_r^*$



(b) Partition of $\mathbf{D}_c^*$



(c) Partition of $\mathbf{D}^*$

$$\begin{bmatrix} 0.87 & 0.02 & 0.42 \\ 0.89 & 0.24 & 0.56 \\ 0.35 & 0.11 & 0.21 \\ 0.80 & 0.33 & 0.43 \\ 0.73 & 0.56 & 0.35 \\ 0.38 & 0.54 & 0.44 \\ 0.01 & 0.81 & 0.54 \\ 0.02 & 0.86 & 0.48 \\ 0 & 0.64 & 0.04 \\ 0.48 & 0.44 & 0.50 \end{bmatrix}$$

(d) Partition values of $\mathbf{U}_{co}$

Figure 4.29: ReSL partitions of co-VAT reordered VOTE dissimilarity images shown in Fig. 4.17

Fig. 4.29(c). The two large co-clusters—the dark blocks in the upper left and lower middle—are the Republicans an Democrats, respectively, that vote along party lines and the votes on which they vote ‘nay’ (recall ‘nay’ is coded as 0).

Interestingly, because this example is not “true” relational data, i.e. the elements of $\mathbf{D}$ are not a measure of dissimilarity, the co-clusters with a small degree of co-clusterness are also informative. These co-clusters represent representatives that vote ‘yea’ on the same votes.

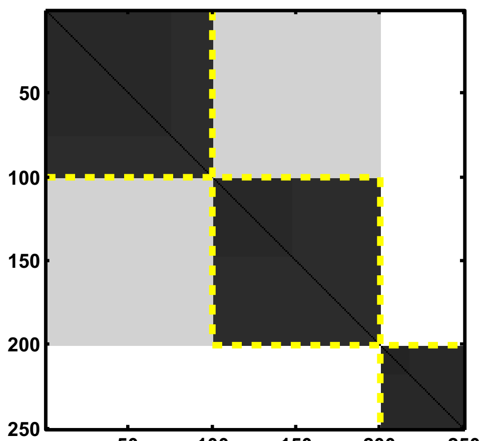
For this reason, I believe that, in this case, the degree of co-clusterness matrix is representative of the type of each co-cluster. High values indicate representatives and the votes on which they vote ‘nay’ together, low values indicate representatives and the votes on which they vote ‘yea’ together, and values near 0.5 indicate mixed groups.

Next, I will examine how co-iVAT can be used as the input for ReSL.

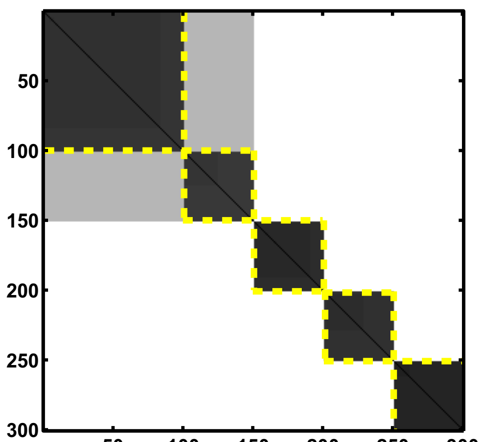
ReSL and co-iVAT

Example 4.3.6. This example presents the ReSL partitions of the data set shown in Figs. 4.15, 4.22, and 4.27. This set represents one of the examples where the idea of the co-cluster is questioned; namely, is there 1 co-cluster in this data set or are there 6? The ReSL degree of co-clusterness matrix, computed on the co-VAT image and shown in Fig. 4.27(e), suggests that there are 6 co-clusters by the 6 large values.

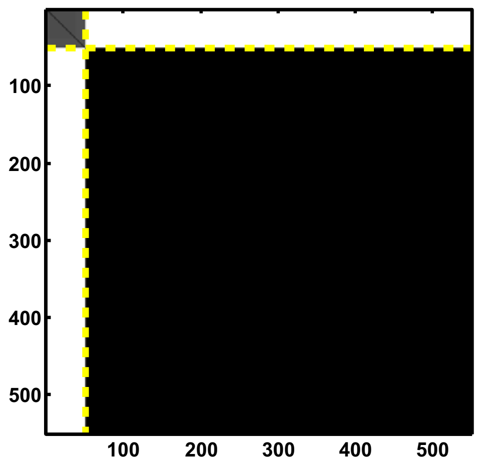
However, using ReSL with the co-iVAT images as input produces a different result. The results are shown in Fig. 4.30. The partitions of the co-iVAT images, $\mathbf{D}_r'^*$ and $\mathbf{D}_c'^*$, are identical to the partitions of the co-VAT images, $\mathbf{D}_r^*$ and $\mathbf{D}_c^*$. However, because the image of $\mathbf{D}_{r\cup c}'^*$ is drastically different from the image of $\mathbf{D}_{r\cup c}^*$, the partition show in view (c) is also very different. There are 2 clusters in to co-iVAT image $\mathbf{D}_{r\cup c}^*$, while there were 3 clusters in the co-VAT image $\mathbf{D}_{r\cup c}^*$. Unlike the co-VAT image, the



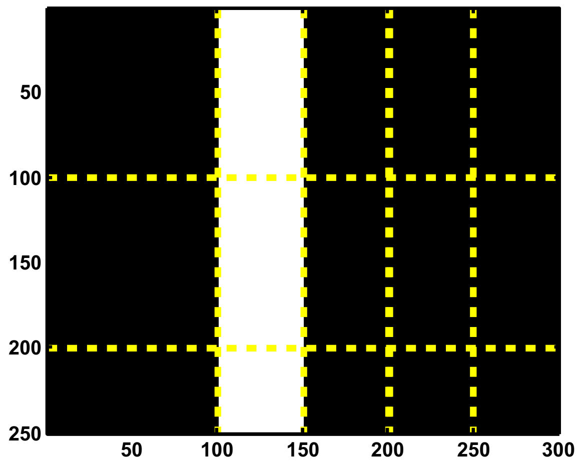
(a) Partition of $\mathbf{D}'^*_r$



(b) Partition of $\mathbf{D}'^*_c$



(c) Partition of $\mathbf{D}'^*_{r\cup c}$



(d) Partition of $\mathbf{D}'^*$

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

(e) Partition values of $\mathbf{U}_{co}$

Figure 4.30: ReSL partitions of co-iVAT reordered dissimilarity images shown in Fig. 4.22

co-iVAT has a visually preferable partition and CLODD is finds it. Last, the partition of $\mathbf{D}'^*$ is shown in view (d) and the degree of co-clusterness matrix is shown in view (e). The partition of degree of co-clusterness matrix support 12 co-clusters. However, note that the 12 co-clusters form a large dark block where row objects $\{1 - 250\}$ are perfectly similar to the column objects $\{1 - 100, 150 - 300\}$. Thus, it is my opinion that this co-iVAT image has 1 co-cluster, which is formed by the 3 row objects clusters and four of the five column object clusters.

4.3.2 Co-clustering Comparisons

There are many co-clustering algorithms, each producing different types of partitions in different types of data. However, ReSL is the *only* clustering algorithm to date that finds all types of clusters within rectangular proximity or relational data. But for the sake of completeness, in this section I make comparisons with the co-clustering algorithm, *spectral co-clustering*, developed by Dhillon [2001].

The spectral co-clustering algorithm considers the induced adjacency matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{bmatrix},$$

where M_{ij} is the weight of the edge that connects vertexes i and j, and $\mathbf{A}$ is the rectangular adjacency matrix. Because I am considering rectangular relational matrices $\mathbf{D}$, $\mathbf{D}$ can easily be converted to an adjacency matrix with the transformation $\mathbf{A} = 1 - \mathbf{D}$. Notice that the induced adjacency matrix $\mathbf{M}$ assumes that the edges between the row objects and column objects (vertexes) have 0-valued weights (no edges). I argue that this assumption is very constrictive and note that ReSL makes no such assumption.

The spectral co-clustering algorithm seeks to find the k co-clusters (groups of row and column objects) that minimize the necessary cut between clusters (sets of vertexes), where a cut is defined as

$$cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} M_{ij}.$$

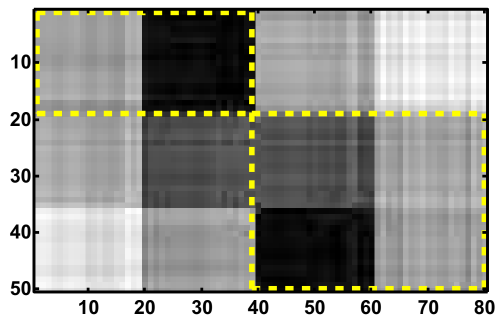
Thus, spectral co-clustering seeks to minimize

$$cut(V_1, V_2, \ldots, V_k) = \sum_{i<j} cut(V_i, V_j), \tag{4.13}$$

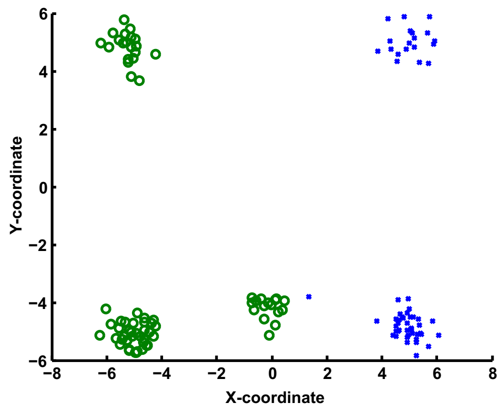
for a given expected number of co-clusters k. I stress that, like many clustering algorithms, spectral-clustering requires the user to input the expected number of co-clusters k. ReSL automatically determines this value, in a sense—it is presented by the degree of co-clusterness matrix. To minimize (4.13), spectral co-clustering uses a *singular value decomposition* (SVD) relaxation-based method. See [Dhillon, 2001] for a detailed description of this algorithm.

Figure 4.31 illustrates the result of spectral co-clustering on the dissimilarity matrix of the data set first presented in Fig. 2.9. Recall that the two bottom clusters on the left and right are composed of both row and column objects, the middle cluster is a set of row objects, and the top two clusters are column objects. Views (a) and (b) show the 2-partition of these data. The yellow-dotted lines in view (a) illustrate the partition of the dissimilarity data, while the plot in view (b) shows the corresponding 2-partition of the object data. Note that the two true co-clusters, as shown by the two dark blocks in the dissimilarity matrix, are partitioned into separate clusters.

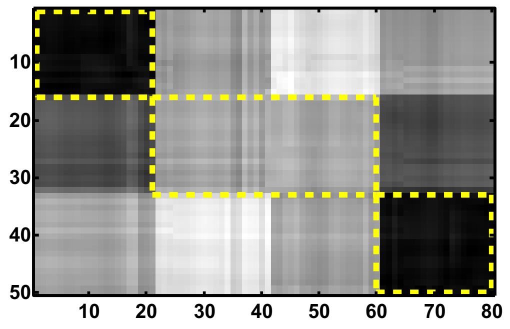
The spectral co-clustering 3-partition of the data set shown in Fig. 4.31(c,d) is, perhaps, more interesting. The two "true" co-clusters comprise two of the clusters and



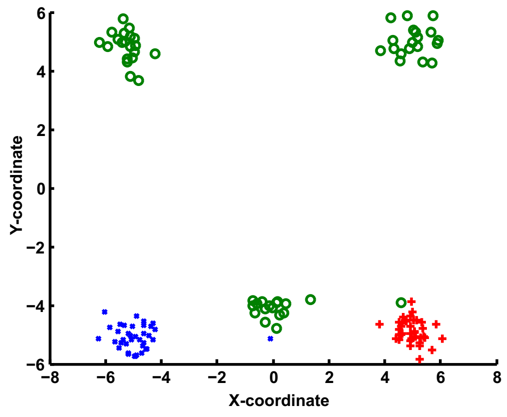
(a) 2-partition, dissimilarity data



(b) 2-partition, object data



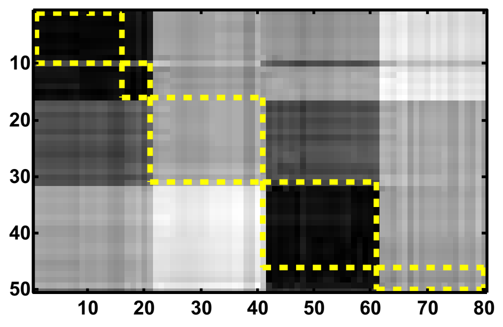
(c) 3-partition, dissimilarity data



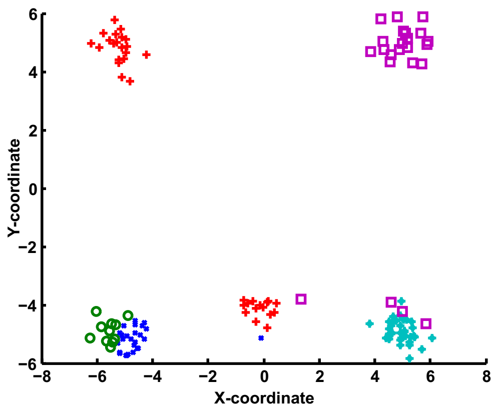
(d) 3-partition, object data

Figure 4.31: Spectral co-clustering partitions of dissimilarity data shown in Figs. 2.9 and 4.25.

the remaining objects—the objects in the middle and top clusters—are grouped in the remaining cluster. In my opinion, this would be the “best” co-cluster partition of this data produced by the spectral co-clustering algorithm. The 4-, 5-, 6-, . . . partitions from spectral co-clustering were unstable and produced wildly varying and “wrong” partitions. The 3-partition of this data is the closest match to the ReSL results for this data set (shown in Fig. 4.25). However, ReSL is able to easily distinguish all 5 of the clusters, including the two co-clusters, the two column-object clusters, and the single row-object cluster. Spectral co-clustering is not only unable to recognize all 5 clusters, but also the closest result is the 3-partition shown in Fig. 4.31(c,d). These



(a) 5-partition, dissimilarity data

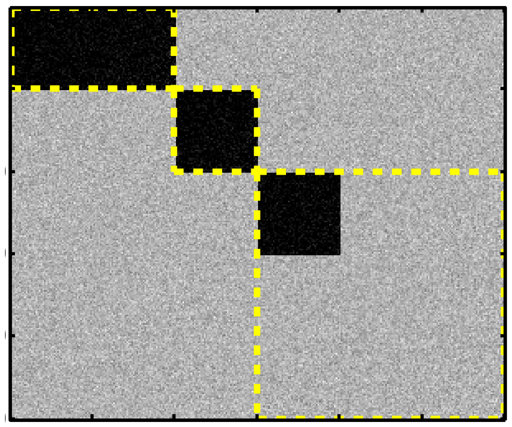


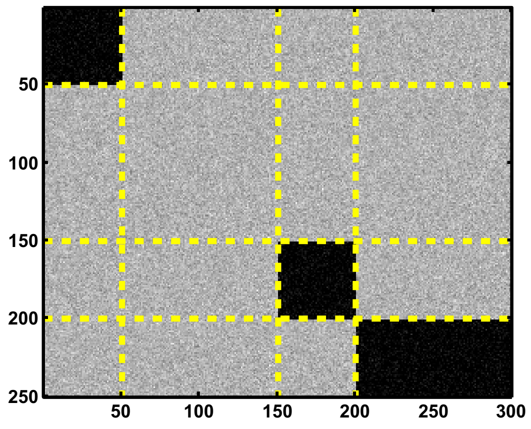
(b) 5-partition, object data

Figure 4.32: Spectral co-clustering 5-partition of dissimilarity data shown in Figs. 2.9, 4.25, and 4.31.

views show that spectral co-clustering is able to partition the two true co-clusters, but the other three are then grouped as one. For a final comparison, the spectral co-clustering 5-partition is shown in Fig. 4.32; clearly, this result is undesirable.

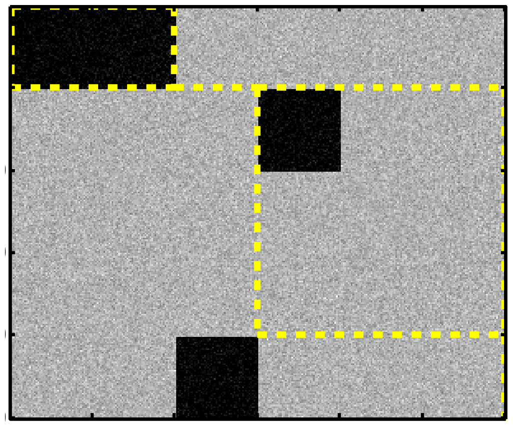
Figure 4.33 demonstrates spectral co-clustering on one of the "pure" relational data sets. The ReSL partitions are shown in Fig. 4.26—I have included the image of the ReSL partition of the rectangular matrix in view (b) of 4.33 for comparison. To reiterate, ReSL partitions the row objects into 4 clusters, the column objects into 4 clusters, and definitively partitions the three dark blocks as three co-clusters, as shown in the degree of co-clusterness matrix in Fig. 4.26(e). The "best" and most stable spectral co-clustering partition—the 3-partition—of these data is shown in Fig. 4.33. This algorithm partitions the three dark blocks into separate co-clusters; however, it also groups the objects that should not belong to a co-cluster into one of the clusters. One might imagine that the spectral co-clustering 4-partition would be more pleasing. However, the 4-partition is unstable (it can be wildly different, depending on the initialization). Figure 4.34 shows two results of the 4-partition produced by spectral co-clustering—one result is "correct" or desirable, the other


(a) Spectral co-clustering partition



(b) ReSL partition

Figure 4.33: Spectral co-clustering 3-partition of dissimilarity data shown in Figs. 2.10 and 4.26. ReSL partition shown in view (b) for comparison.


(a) 4-partition 'A'

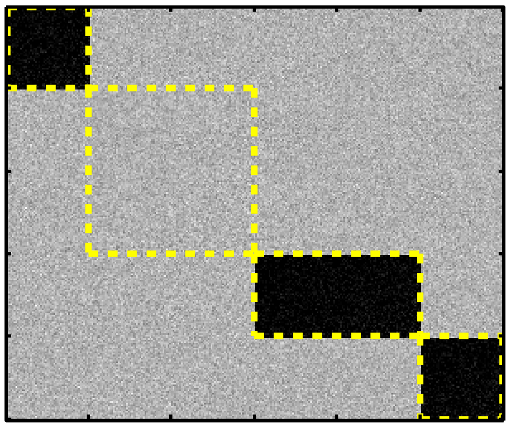
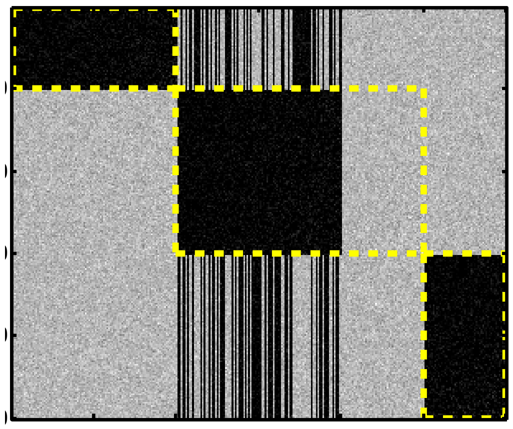

(b) 4-partition 'B'

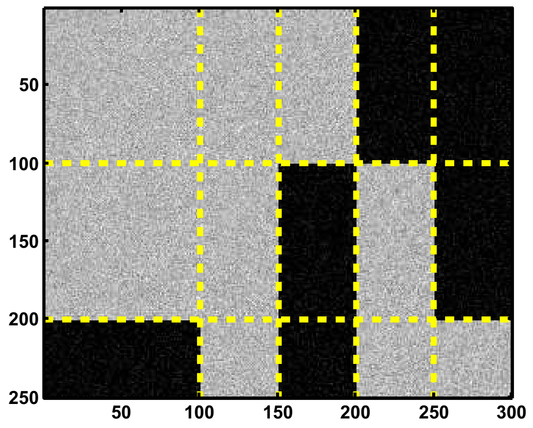
Figure 4.34: Spectral co-clustering 4-partitions of dissimilarity data shown in Figs. 2.10, 4.26, and 4.33.

result is not. ReSL, by comparison, produces the "correct" result, every time.

Finally, Figs. 4.35 and 4.36 demonstrate spectral co-clustering on the "pure" relational data sets first presented in Figs. 4.15 and 4.16. I show the ReSL partition of the rectangular dissimilarity matrix for each example for comparison. In each of these examples, spectral co-clustering produced unstable partitions (very different partitions, depending on initialization) at all values of k, the number of co-clusters. I chose some of the more successful results to show in Figs. 4.35 and 4.36. However,



(a) Spectral co-clustering partition



(b) ReSL partition

Figure 4.35: Spectral co-clustering 3-partition of dissimilarity data shown in Figs. 4.15 and 4.27. ReSL partition shown in view (b) for comparison.

even in these more successful cases, spectral co-clustering is unable to partition the dark blocks in the (what I consider to be) desirable way that ReSL does.

In summary, I believe that ReSL is more effective than spectral co-clustering at solving the clustering in rectangular data problem because:

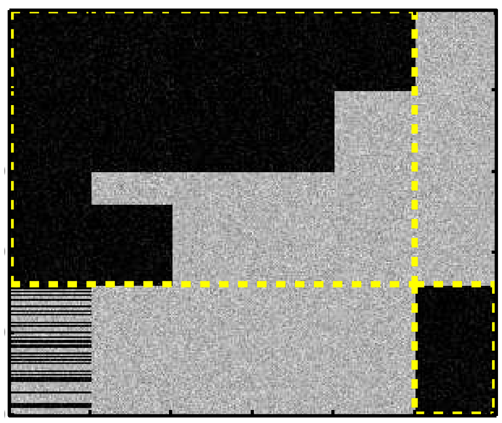
(i) It automatically determines the number of clusters;

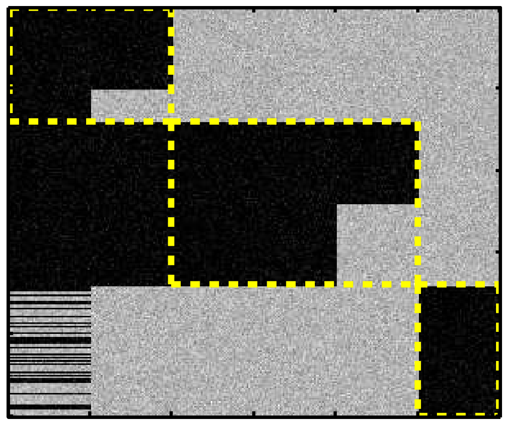
(ii) It produces and recognizes all four types of clusters;

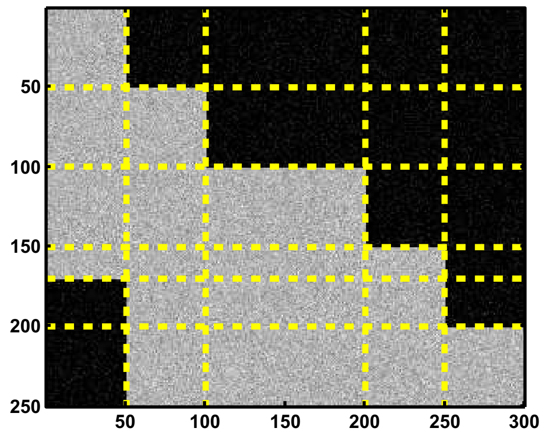
(iii) The partitions ReSL produces in the "pure" relational data examples are more desirable.

4.3.3 Perspectives on ReSL

The examples shown here demonstrate that ReSL is effective at partitioning rectangular data, constructed both from "pure" relational data and Euclidean relations. Examples 4.3.1 and 4.3.5 illustrate data sets where each type of rectangular data cluster—clusters in the row objects, clusters in the column objects, clusters in the


(a) Spectral co-clustering 2-partition


(b) Spectral co-clustering 3-partition



(c) ReSL partition

Figure 4.36: Spectral co-clustering partitions of dissimilarity data shown in Figs. 4.16 and 4.28. ReSL partition shown in view (c) for comparison.

union of the objects, and co-clusters—is clearly present. And ReSL is successful in finding the preferred partition for each of these cluster types.

Examples 4.3.3 and 4.3.4 prove challenging because the definition of a co-cluster is uncertain. In each of these examples, there is a group of individual co-clusters (clusters composed of both row and column objects) which have some relation to the other co-clusters via either the similarity among row objects or similarity among column objects. For example, the co-cluster composed of row objects $\{100-200\}$ and column objects $\{150-200\}$ in Fig. 4.27(d) is similar to the co-cluster composed of row objects $\{200-250\}$ and column objects $\{150-200\}$ and the co-cluster composed of row objects $\{100-200\}$ and column objects $\{250-300\}$. However, notice that the row objects $\{200-250\}$ are dissimilar to the column objects $\{250-300\}$. Thus, the question remains as to whether these three co-clusters are truly only one co-cluster. It is my opinion that this situation is analogous to object-data that has substructure within its clusters. However, because object data does not exist for the rectangular data shown in Examples 4.3.3 and 4.3.4, it is impossible to visually verify whether this is the case. In Appendix B, I briefly explore the question: What is a co-cluster?

I also demonstrated ReSL on real data—the voting records of the 435 members of the U.S. House of Representatives. Again, this data is not truly relational as the elements of $\mathbf{D}$ represent the votes themselves. Thus, a value of 0 does indicate perfect dissimilarity between a member and their vote; it represents a 'nay' vote. Despite this, I believe that ReSL was effective in showing the relationship among the members and the votes on which they agree. I imagine the ReSL could be useful in this case to the members to determine who their allies are in particular issues.

Last, I showed an example of ReSL using co-iVAT as the reordering algorithm. This example further examined the question of, 'what is a co-cluster?' As the co-iVAT examples showed, the co-iVAT transform produces rectangular data images of which the cluster structure is more visually apparent. Because CLODD is used to partition the various matrices produced by co-VAT, the strong edges that are usually present in co-iVAT images allows CLODD to be that much more effective at finding the preferred partition. However, as we saw in many of these examples, co-iVAT does not always produce the most pleasing results (or, perhaps, the desired results). Thus, I recommend clustering both the co-VAT and co-iVAT images. Because it is shown that the iVAT transform has the same computational complexity as VAT, this result carries over to co-VAT and co-iVAT. Thus, very little additional computational time is required to consider both.

Additional examples using CLODD, the alternate co-VAT reordering method, co-iVAT, and ReSL and presented in Appendix . In the next chapter, I switch gears and examine a different type of data: ontologies.

Chapter 5

Clustering in Ontologies

5.1 Ontological Self-Organizing Map (OSOM)

Figure 5.1 is a block diagram of the OSOM training algorithm. The inputs are the ontological data and the pair-wise term similarity matrix (see Section 2.4). The OSOM itself operates very much like a conventional SOM: i) a random test signal is chosen, ii) the winning prototype is selected, iii) all prototypes are moved towards the test signal according to a predefined network topology. Section 5.1.2 describes the training procedure in more detail and also proposes a batch version of the OSOM, which is based on the batch SOM [Kohonen, 2001].

Section 5.1.3 describes how I utilize the OSOM to produce cluster visualization of the ontological data. The visualization method maps the *ontological profiles* (the OSOM prototypes) of the OSOM network to a two-dimensional toroidal grid (although, any predefined network topology could be chosen). Cluster tendency is shown by the relations between neighboring ontological prototypes on the grid, which are displayed as gray levels—black represents *no relation* and white represents *highly related*.

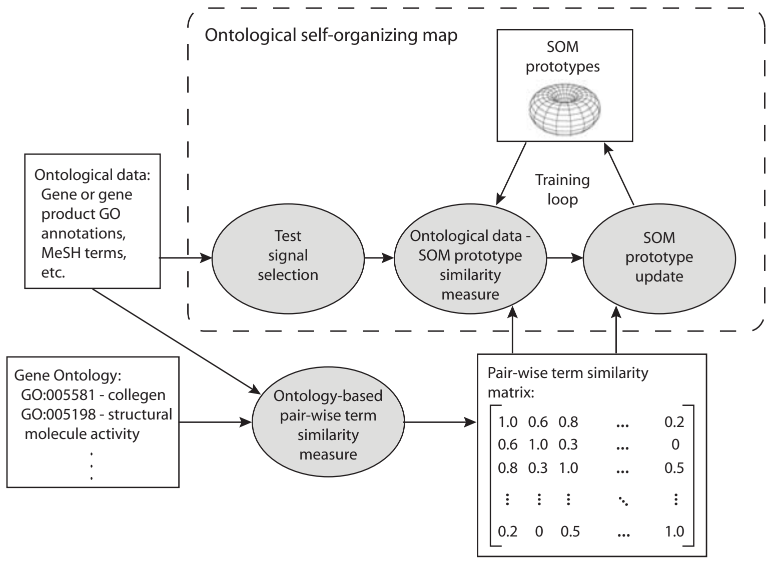

Ontological self-organizing map
SOM prototypes
Training loop
Ontological data: Gene or gene product GO annotations, MeSH terms, etc.
Test signal selection
Ontological data - SOM prototype similarity measure
SOM prototype update
Gene Ontology: GO:005581 - collegen GO:005198 - structural molecule activity
Ontology-based pair-wise term similarity measure
Pair-wise term similarity matrix:
1.0 0.6 0.8 ... 0.2
0.6 1.0 0.3 ... 0
0.8 0.3 1.0 ... 0.5
0.2 0 0.5 ... 1.0

Figure 5.1: OSOM training block diagram.

Summarization of each ontological prototype (e.g. gene or gene product cluster) is achieved as a direct result of my formulation of the OSOM. The OSOM prototypes are represented by a vector of weights, where each element of the weight vector is associated with a term from the corpus. The value of these weights are the memberships of the associated terms in the description of the ontological prototype. Thus, the summarization of each prototype is the term(s) with the largest corresponding weight vector element(s). If the ontological data are genes represented by their GO annotations, then this summarization describes the shared function of the groups of genes. Section 5.1.4 describes my summarization method in more detail.

5.1.1 Generalized Outer Product Similarity

I use a distance measure in this section, the *generalized outer product* (GOP), which has not previously been used for GO-based gene similarity. This measure was discovered in our work in [Sledge et al., 2009c,b], which involves generalizing cluster validity indexes to relational data . Essentially, the GOP distance is the well known A-norm

($||\vec{x} - \vec{y}||_A$) generalized to relational data. The authors of [Hasenfuss and Hammer, 2007] use this distance in their relational SOM.

First, assume that $\vec{g}_i$ is the binary vector representation of G_i and $D_{kl} = 1 - R_{kl}$ is the normalized distance between the kth and lth terms. The GOP distance is

$$d^{(GOP)}(G_i, G_j) = \tilde{g}_i^T \mathbf{D} \tilde{g}_j - 0.5 \tilde{g}_i^T \mathbf{D} \tilde{g}_i - 0.5 \tilde{g}_j^T \mathbf{D} \tilde{g}_j, \tag{5.1}$$

where $\tilde{g} = \frac{\vec{g}}{||\vec{g}||_1}$ and $\mathbf{D} = 1 - \mathbf{R}$.

Assume $D_{lm} = ||\vec{T}_l - \vec{T}_m||_2$, where $\vec{T}_l$ and $\vec{T}_m$ are the hypothetical object-space representations of the lth and mth terms (i.e. if one could represent the GO term, GO:0016740-transferase activity, as a set of coordinates in some space, which to my knowledge is impossible). If we represent two objects as the weighted sums of the vectors $\vec{T}$,

$$\vec{v}_i = \sum_l (\tilde{g}_i)_{l=1}^{N_T} \vec{T}_l,$$

$$\vec{v}_j = \sum_m (\tilde{g}_j)_{m=1}^{N_T} \vec{T}_m,$$

then it can be shown that

$$||\vec{v}_i - \vec{v}_j||_2^2 = \tilde{g}_i^T \mathbf{D} \tilde{g}_j - 0.5 \tilde{g}_i^T \mathbf{D} \tilde{g}_i - 0.5 \tilde{g}_j^T \mathbf{D} \tilde{g}_j.$$

Therefore, it follows that $d^{(GOP)}(G_i, G_j) = ||\vec{v}_i - \vec{v}_j||_2^2$, where the vectors $\vec{v}$ are essentially weighted averages of the object-space representations of the hypothetical object-space terms $\vec{T}$. Hence, $\vec{v}_i$ is the object-space representation of the ith gene $\vec{g}_i$ (in clustering this is called a medoid) and $\vec{v}_j$ is the object-space representation of the jth gene $\vec{g}_j$.

Reference [Sledge et al., 2009b] contains a detailed discussion on the derivation of this distance measure in the context of relational clustering and medoids. I will show in Section 5.1.5 that the OSOM and the standard SOM are related when the $d_{(GOP)}$ distance measure is used.

I combine all the terms from a set of gene products into one large set and compute a pair-wise GO-term similarity matrix $\mathbf{R}$ using the Lin information-theoretic similarity measure [Lord et al., 2003] in Eq. (2.30). The Lin similarity measure is effective at computing the similarity of two GO terms because it considers both the relative closeness of two terms as well as the depth (specificity) of the terms in the hierarchy. The $GPD194$ data set [Keller et al., 2004] contains 64 total unique GO terms; thus, the Lin-based similarity matrix $\mathbf{R}$ is 64×64. The pre-computed similarity matrix allows one to quickly compute similarities by casting many of the operations in the OSOM as matrix-vector multiplications.

5.1.2 Ontological Self-Organizing Map Definition

The self-organizing map is a two-layer lateral feedback neural network that topologically maps itself to the training data. The network structure is often set to a two-dimensional square, toroidal, or hexagonal grid, where each network node, or prototype, is laterally connected to its neighbors. The network learning algorithm is as follows:

1. Randomly draw a sample from the training data, $\vec{x}_d$.

2. Find closest SOM prototype p according to a chosen distance metric,

$$p = \arg\min_i \{||\vec{x}_d - \vec{a}_i^{(old)}||\}. \tag{5.2}$$

3. Update SOM prototypes by

$$\vec{a}_i^{(new)} = \vec{a}_i^{(old)} + \epsilon(t) \cdot h_{ip} \cdot (\vec{x}_d - \vec{a}_i^{(old)}), \tag{5.3}$$

   where $\epsilon(t)$ is the learning rate and h_{ip} is the neighborhood function defined as

$$h_{ip}(t) = \exp\left(-\frac{|\vec{a}_i - \vec{a}_p|^2}{\sigma^2(t)}\right), \tag{5.4}$$

   where $\vec{a}_i$ is the location of the SOM prototype in the predefined neighborhood (e.g. square or hexagonal grid).

This algorithm is repeated until a maximum number of iterations or convergence is reached. Typically, the learning rate $\epsilon(t)$ and the width of the neighborhood function $\sigma^2(t)$ are reduced during iteration, with the effect that late iterations are only applying small updates to network prototypes local to the winning prototype p.

Prototype representation

The algorithm I propose as the OSOM is an adaption of the standard SOM to ontological data. First, I construct an ontological weight vector for each node in the OSOM grid. This weight vector is a fuzzy membership representation of all the terms present in the training data. For example, the $GPD194$ data set contains a total of 64 terms among all the gene products combined; thus, the OSOM weight vector has a length of 64. Each weight vector element is associated with one term and the value of the weight is the membership of the associated term in the description of the ontological prototype. I denote the OSOM weight vectors as $\vec{w}_i \in [0, 1]^{N_T}$.

Second, I replace the distance metric in step 2 of the SOM with a similarity measure. The measures I use are vector-matrix multiplication-based operations that are simple extensions of the measures described in Section 2.4.1, [Keller et al., 2004], and [Sledge et al., 2009b]. What makes the OSOM similarity measures different is that they are the similarity between an OSOM protoype and a gene or gene product; the similarity measures in Section 2.4.1 are for two genes or gene products. In practice, one could choose any similarity measure that measures the similarity of two sets of terms; there are many measures that fit this description. However, I recommend using similarity measures that perform some aggregation on the pair-wise similarity matrix **R**. Set-based similarity measures such as the Cosine and Jaccard index are ill-suited to this problem (see [Keller et al., 2004] for information on this topic). I adapt gene-gene similarity measures, such as the average in Eq. (2.32), for use with the OSOM:

- Generalized outer product (GOP):

  $$s^{(GOP)}(\vec{w}_i, \vec{g}_j) = 1 - \tilde{w}_i^T \mathbf{D} \tilde{g}_j + 0.5 \tilde{w}_i^T \mathbf{D} \tilde{w}_i + 0.5 \tilde{g}_j^T \mathbf{D} \tilde{g}, \tag{5.5}$$

  where

  $$\tilde{w}_i = \frac{\vec{w}_i}{|\vec{w}_i|},$$
  $$\tilde{g}_j = \frac{\vec{g}_j}{|\vec{g}_j|},$$

  and $\mathbf{D} = 1 - \mathbf{R}$.

- Average (AVG):

  $$s^{(AVG)}(\vec{w}_i, \vec{g}_j) = \frac{\vec{w}_i^T \mathbf{R} \vec{g}_j}{|\vec{w}_i||\vec{g}_j|} \tag{5.6}$$

- Ordered Weighted Average (OWA) [Yager and Kacprzyk, 1997]:

  $$\vec{l} = (\mathbf{R}\vec{w}_i). * \vec{g}_j,$$

where $.*$ represents element-by-element multiplication. The vector l is then sorted in descending order, $l_{(1)} > l_{(2)} > ... > l_{(N_T)}$, and the OWA similarity is computed by,

$$s^{(OWA)}(\vec{w}_i, \vec{g}_j) = \frac{1}{N_T^2} \sum_{k=1}^{N_T} b_k l_{(k)}, \tag{5.7}$$

where N_T is the number of terms and b_k is the weight of the kth term in the OWA. In this paper we use $b_k = 1,\ k \leq 4$ and $b_k = 0,\ k > 4$, i.e., the average of at least 4 pair-wise similarities. However, one could certainly choose a different set of b_k's, e.g. $b_1 = 1$ and $b_k = 0,\ k > 1$ is the maximum pair-wise similarity between $\vec{w}_i$ and the gene product term vector $\vec{g}_j$.

$s^{(AVG)}$ and $s^{(OWA)}$ are adaptations of the soft similarity measures in [Keller et al., 2004], which were developed specifically for ontologies.

$s^{(GOP)}$ is adapted from relational clustering and the distance measure $d^{(GOP)}$. An important point is that with the $s^{(GOP)}$ similarity measure, the OSOM can be shown to be equivalent to Kohonen's SOM (see the first example in Sec. 5.1.5). However, unlike the SOM, the OSOM can also be used with ontological data.

Prototype update

The OSOM can use the standard weight vector update Eq. (5.3) of the SOM by substituting $\vec{g}_d$ for $\vec{x}_d$,

$$\vec{w}_i^{(new)} = \vec{w}_i^{(old)} + \epsilon(t) \cdot h_{ip} \cdot (\vec{g}_d - \vec{w}_i^{(old)}). \tag{5.8}$$

Recall that $\vec{g}$ is binary; hence, this update simply moves the prototype towards the corresponding corner of the N_T-dimensional hypercube. This update, however, ignores the term-term similarities.

I also replace the standard form of the weight vector update equation with a similarity-based update. In order to create an similarity-based update equation, Idefined two axioms:

1. At each iteration, the weight vector elements that correspond to the terms in the test signal $\vec{g}_d$ must increase, as in Eq. (5.8).
2. At each iteration, the weight vector elements that are similar to the terms in $\vec{g}_d$, as evidenced by $\mathbf{R}$, must also increase.

With these axioms in mind, I created the following update equation

$$\vec{w}_i^{(new)} = \vec{w}_i^{(old)} + \epsilon(t) \cdot h_{ip}(t) \cdot \left(F(\mathbf{R}, \vec{g}_d) - \vec{w}_i^{(old)}\right), \quad \forall i, \tag{5.9}$$

where p denotes the closest OSOM prototype to the randomly chosen training vector $\vec{g}_d$ and where $(F(\mathbf{R}, \vec{g}_d) - \vec{w}_i^{(old)})$ is the update operator. As shown below, the update operator is computed from the columns of the similarity matrix that correspond to non-zero elements of the training vector $\vec{g}_d$. These columns of the similarity matrix represent the similarity between the terms in $\vec{g}_d$ and all other terms (e.g. R_{ij} is the similarity of the ith and jth terms). Hence, the update operator, $F(\mathbf{R}, \vec{g}_d)$, computes a row aggregation on the columns of the similarity matrix $\mathbf{R}$ that correspond to the terms in the training vector $\vec{g}_d$. The operator F can be modeled after any aggregation operator [Klir and Yuan, 1995]; e.g., one can define F as one of the following:

- Average (AVG):
$$F^{(AVG)}(\mathbf{R}, \vec{g}_d) = \frac{\mathbf{R}\vec{g}_d}{|\vec{g}_d|}. \tag{5.10}$$
- Maximum (MAX):
$$F_k^{(MAX)}(\mathbf{R}, \vec{g}_d) = \max_i \{R_{ki}\}, \tag{5.11}$$

where $i = \{l \in \mathbb{N} | l \leq N_T; (\vec{g}_d)_l = 1\}$, $k = 1, ..., N_T$, and R_{ki} is the i-th column of the k-th row of the similarity matrix $\mathbf{R}$.

The operator chosen for F determines the convergence behavior of the values of the prototype weight vectors, $\{\vec{w}_i\}$. For example, $F^{(AVG)}$ causes the weight vectors to have maximum values around 0.5, as the operator averages the similarity values for all terms in the training vectors. Contrastively, $F^{(MAX)}$ causes the maximum weight vector values to tend to a value of 1, as there are exactly $|\vec{g}_d|$ terms equal to 1 in the matrix-vector multiplication $\mathbf{R}\vec{g}_d$ (each term in $\vec{g}_d$ has similarity of 1 to itself). Simply put, $F^{(MAX)}$ pushes the OSOM prototypes towards the terms present in $\vec{g}_d$ and, additionally, pushes the prototypes towards all terms represented in $\mathbf{R}$ that are similar to any one of the terms in $\vec{g}_d$. Both (5.10) and (5.11) are variations of one form of the generalized mean, where $F^{(AVG)}$ is M_1 and $F^{(MAX)}$ is M_∞. The ith element of M_p is

$$M_p(\mathbf{R}, \vec{g}_d)_i = \left(\frac{1}{|\vec{g}_d|} \sum_{j=1}^{N_T} (R_{ij}(\vec{g}_d)_j)^p \right)^{1/p} .$$

Algorithm 5.1.1 outlines the standard OSOM algorithm. The parameters, such as the learning rates and maximum iterations, are set according to the problem (viz. just like the original SOM, use what works for you). For this paper, I use a toroidal grid-based network as this grid topology does not experience the edge-effects that a standard square-grid does. The learning rates are set to $\{\epsilon_0 = 0.5, \epsilon_f = 0.005\}$, the widths of the lateral influence function in eq.(5.4) are $\{\sigma_0 = N_{net} + 1, \sigma_f = 0.1\}$, and the maximum number of iterations is $t_{max} = 2,000$. The width of the network N_{net} is adjusted depending on the size of the data set, in this case the number of genes or gene products. The illustrative results in this section use a 8×8 toroidal network topology (64 prototypes) to map the 194 gene products in $GPD194$. I chose a toroidal grid because the neighborhood of each prototype is consistent, thus avoiding the well-known boundary effects seen in square-grid network topologies.

Algorithm 5.1.2 outlines the batch version of the OSOM. The strength of the batch SOM, in general, is that it is proven to converge in a finite number of steps [Cottrell et al., 2006]. In Section 5.1.5, I compare the different forms of the OSOM and describe the resulting trained networks.

Algorithm 5.1.1: Ontological Self-Organizing Map

```
Data: $\vec{g}_j, j = 1, \ldots, N_G$ where $\vec{g}_j$ is the $j$-th vector of the training data.
Randomly initialize OSOM prototype weight vectors $\vec{w}_i \in [0,1]^{N_T}$.
$t \leftarrow 0$
while $t < t_{max}$ do
    Randomly draw a single training data vector $\vec{g}_d$.
    Find closest prototype, $p = \arg\max_i S(\vec{w}_i, \vec{g}_d)$.
    Update prototypes weight vectors with Eq.(5.9).
    $\sigma(t) = \sigma_0(\sigma_f/\sigma_0)^{t/t_{max}}$
    $\epsilon(t) = \epsilon_0(\epsilon_f/\epsilon_0)^{t/t_{max}}$
    $t \leftarrow t + 1$
```

Algorithm 5.1.2: Batch Ontological Self-Organizing Map

```
Data: $\vec{g}_j, j = 1, \ldots, N_G$
Randomly initialize OSOM prototype vectors $\vec{w}_i \in [0,1]^{N_T}$, $\sum_{j=1}^{N_T} w_{ij} = 1$.
$t \leftarrow 0$
while $t < t_{max}$ do
    $\vec{w}'_i = \{0\}^{N_G}, \forall i$
    for $j = 1$ to $N_G$ do
        Find closest prototype, $p = \arg\max_i S(\vec{w}_i, \vec{g}_j)$.
        $\vec{w}'_i = \vec{w}'_i + h_{ip}(t) \cdot \vec{g}_j, \forall i$
    $\vec{w}_i = \frac{\vec{w}'_i}{|\vec{w}'_i|}, \forall i$
    $\sigma(t) = \sigma_0(\sigma_f/\sigma_0)^{t/t_{max}}$
    $t \leftarrow t + 1$
```

5.1.3 Cluster Visualization

The visualization method I propose is composed of two distinct steps. First, the objects (e.g. gene products, articles, etc.) are mapped to the trained OSOM network

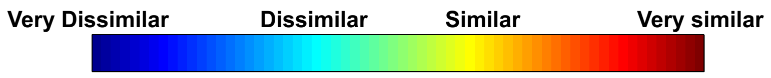


Figure 5.2: Colormap used in visualizations shows relative similarity between network prototypes.

by the nearest prototype rule—for each object $\vec{g}$, find the best match prototype with $p = \arg\max_i S(\vec{w}_i, \vec{g})$. The prototype p is then annotated with the object information of $\vec{g}$ (e.g. the gene product id, the GO annotations). This groups similar objects (gene products) into cluster-like arrangements, where each OSOM prototype essentially represents a cluster (sometimes an empty cluster). Second, the similarity between neighboring OSOM prototype nodes is mapped into a gray-scale or color image [Kaski and Kohonen, 1996]—for this paper, red indicates *very similar*, blue indicates *very dissimilar*. The colormap used in this paper is shown in Fig. 5.2. Figure 5.3 illustrates this mapping for $GPD194$ using the similarity $s^{(GOP)}$, eq.(5.5), and the batch OSOM update. The red regions correspond to groups of similar gene products, while the blue regions show the boundaries between dissimilar regions. Please note that because of the toroidal network topology, the top and bottom, as well as the sides, of the images in Fig. 5.3 wrap around. Also, because $GPD194$ contains multiple (but different) sequences of each gene product, the gene product labels (e.g. FGFR1) can appear in more than one place on the OSOM map.

I compute the similarity between nodes with a generalized outer-product operator, as in [Hasenfuss and Hammer, 2007],

$$s(\vec{w}_i, \vec{w}_j) = 1 - \sqrt{\tilde{w}_i^T \mathbf{D} \tilde{w}_j + 0.5 \tilde{w}_i^T \mathbf{D} \tilde{w}_i + 0.5 \tilde{w}_j^T \mathbf{D} \tilde{w}_j}, \tag{5.12}$$

$$\tilde{w} = \frac{\vec{w}}{|\vec{w}|},$$

and $\mathbf{D} = 1 - \mathbf{R}$. The reason that I use a square-root in this calculation is because

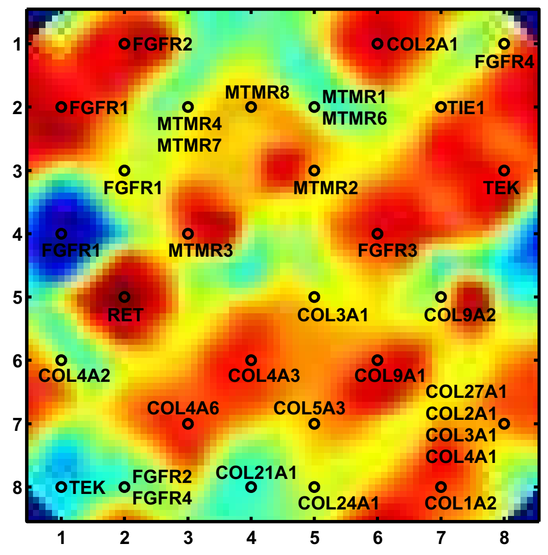


Figure 5.3: Batch OSOM network mapping of $GPD194$ using $s^{(GOP)}$.

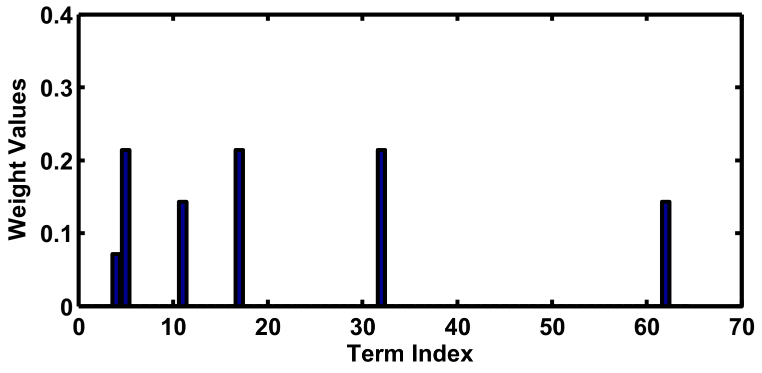
the colormap appears more linear with distance (recall that the GOP distance is equivalent to squared Euclidean distance), which I have found to be more effective. One could include the square-root in the $s^{(GOP)}$ calculation if desired, but, because $s^{(GOP)}$ is only used to find the closest prototype to the random test signal, the square root is unnecessary. The similarity in (5.12) is calculated between each connected node of the OSOM network. Thus, for the toroidal grid each prototype node has four surrounding pixels which correspond to its relation to its neighboring nodes. The colormap is set such that red corresponds to $\max_{\forall i, \forall j}\left[s(\vec{w}_i, \vec{w}_j)\right]$ and dark-blue corresponds to $\min_{\forall i, \forall j}\left[s(\vec{w}_i, \vec{w}_j)\right]$ for a given network. The color-grid is then up-sampled by cubic interpolation to achieve a visually-pleasing map.

As a result of this coloring scheme, regions that are red represent groups of similar objects, while blue and cyan regions signify boundaries or objects that are dissimilar to the surrounding groups. In addition, the degree of similarity can be inferred from the color-intensity of the regions. For example, in Fig. 5.3, the red islands indicate groups of similar prototypes—e.g. the red region centered at (4,6) is a group of *collagens*—while the surrounding blue-cyan regions signify boundaries. Additionally, the dark-blue region at the map location (1,4) (labeled FGFR1) denotes a dissimilar gene product.

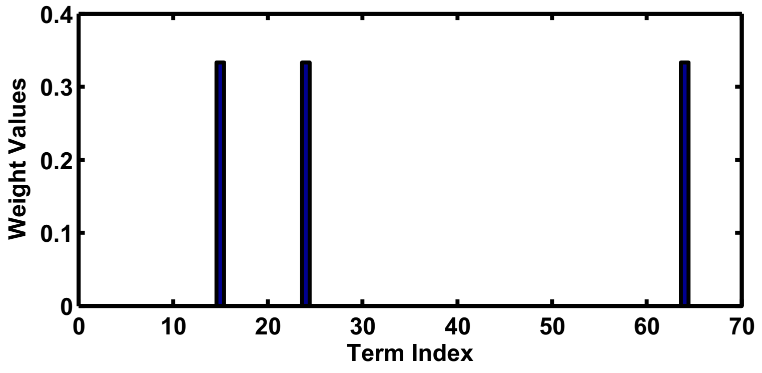
The three $GPD194$ families can be seen in Fig. 5.3 as spatially grouped gene products. The *collagen alpha chains* (COL) are located in the lower part of the image, with one group mapped to the top right. Recall that the grid is toroidal; hence, these regions are connected. The *myotubularins* (MTMR) are located at the upper-center. Lastly, the *receptor precursors* (FGFR, TEK, TIE, RET) are scattered throughout the red islands on the upper-left and upper-right, with a group also in the lower-right. Note that the MTMR island is connected to the FGFR islands. This shows that these gene products are related to a degree. These relations are corroborated by other work with these gene products [Popescu et al., 2004].

5.1.4 Cluster Summarization

Cluster summarization, i.e. the potential output of a CW engine, of the ontological prototypes is achieved by examining the OSOM prototype weight vectors. For the case of genes or gene products, this summarization is a functional summarization of each group. The ontological content of each OSOM prototype is represented by a weight vector, as discussed in Section 5.1.2. Each element of the weight vector can be viewed as the relative influence of a specific annotation in defining the profile of its associated OSOM prototype. Thus, high values in a weight vector signify a high likelihood that the objects mapped to a location are annotated by the associated term(s). I define the most representative term (MRT) of an ontological prototype as the term that has the highest associated weight in the OSOM prototype weight vector. If there is more than one maximum weight vector element, then the MRT is defined as the term with the highest information-content [Popescu et al., 2004].



(a) Weight vector of (2,8)



(b) Weight vector of (1,6)

Figure 5.4: Prototype weight vectors for map locations (2,8) and (1,6) in Fig. 5.10(a).

This provides a simple linguistic output for a potentially complex organization of the ontological description of groups of genes.

In general, the OSOM weight vectors represent the set of linguistic descriptions of the mapped objects because each element of the weight vector is the strength of an ontology term. The relative value of each vector element is the degree to which an ontology term or word describes the objects mapped to a specific location. For example, Fig. 5.4 illustrates the prototype weight values for the map locations (2,8) and (1,6) in Fig. 5.3. For the batch OSOM the prototypes are normalized; hence, only the relative values within each prototype are important.

The OSOM weight vectors can be used to construct a linguistic proposition about each map location and the gene products that are mapped to that location. Take, for example, the network prototype weight vectors plotted in Fig. 5.4. The gene products mapped to prototype (2,8) are the *collagens* indexed 168-184. The gene products mapped to prototype (1,6) are the *receptor-precursors* indexed 76, 93, and 96. View (a) shows that the weight vector of prototype (2,8) has 6 non-zero elements. These non-zero elements are the membership of the respective terms that define that prototype; thus, they can be used as an input to a fuzzy-rule system which computes a linguistic summarization. Figure 5.5 shows the rule base for the example I show here. Note that the ordinate of the output is a linguistic proposition in the form of a hedge. I also add the ontology from which each summarizing term comes (i.e. molecular function, biological process, or cellular component) to the linguistic summarization, shown in **bold** in this example.

The output linguistic proposition for (1,6) is: The *collagen* gene products indexed 168-184 are summarized by the following perceptions: i) the **molecular function** is MOSTLY *extracellular matrix structural constituent*; ii) the **cellular component** is MOSTLY *collagen type IV*; iii) the **biological process** is MOSTLY *extracellular matrix organization.* For this example the linguistic propositions are redundant because the weight values are all equal. However, for the gene products mapped to location (2,8), the weight values are not equal. Thus, the linguistic outputs define the relative strength of each term in the proposition. For example: The *receptor-precursors* indexed 76, 93, and 96 are summarized by the following perceptions: i) the **molecular function**s are SLIGHTLY *protein serine/threonine kinase activity*, MOSTLY *protein tyrosine kinase activity*, SOMEWHAT *receptor activity*, MOSTLY *ATP binding*, and SOMEWHAT *transferase activity*; ii) the **biological process** is SOMEWHAT *protein amino acid dephosphorylation.*

Figure 5.6 shows the MRTs for a zoomed-in portion of the trained OSOM network

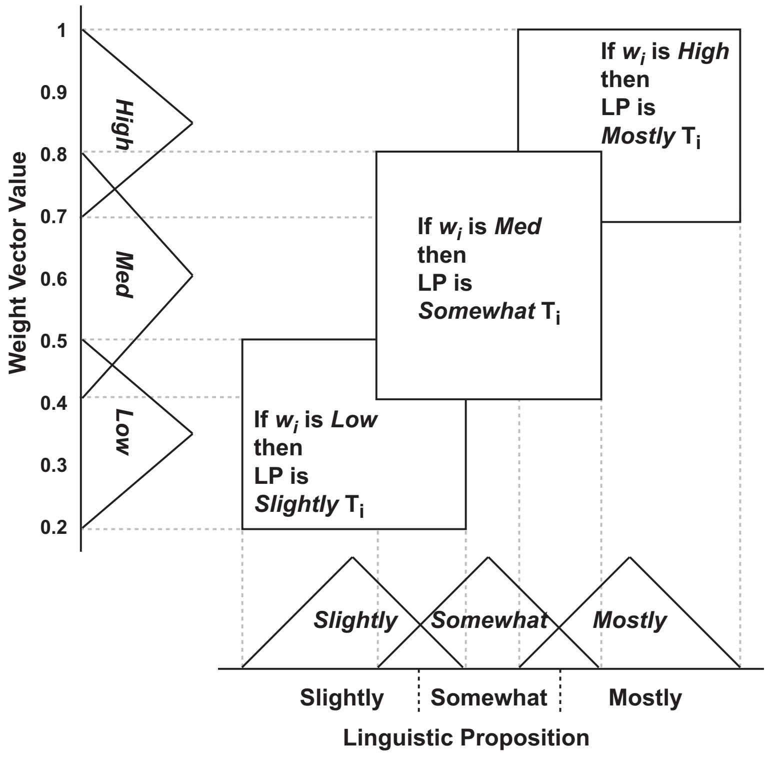

Weight Vector Value
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
High
Med
Low
If w_i is *High* then LP is *Mostly* T_i
If w_i is *Med* then LP is *Somewhat* T_i
If w_i is *Low* then LP is *Slightly* T_i
Slightly
Somewhat
Mostly
Slightly
Somewhat
Mostly
Linguistic Proposition

Figure 5.5: Fuzzy rule-based system — input is normalized prototypes of trained OSOM network. T_i indicates the associated GO term description for prototype element w_i.

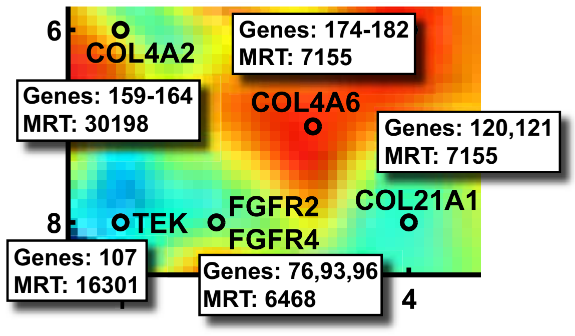


Figure 5.6: Zoomed in view of lower-left portion of Fig. 5.3 that shows mapping of $GPD194$ gene products and most-representative-terms (MRT) of each location

shown in Fig. 5.3. Table 5.1 contains the MRTs for all prototypes in the trained OSOM network shown in Fig. 5.10(a).

Table 5.1: **Most representative terms** (MRTs) of OSOM network trained on $GPD194$ — full map shown in Fig. 5.10(a).

OSOM Index	GO ID	GO Definition
Receptor Precursors		
(2,1)	GO:0016740	transferase activity
(1,2)	GO:0016740	transferase activity
(2,3)	GO:0004872	receptor activity
(1,4)	GO:0004872	receptor activity
(8,1)	GO:0016740	transferase activity
(7,2)	GO:0016740	transferase activity
(8,3)	GO:0016740	transferase activity
(6,4)	GO:0016740	transferase activity
(2,5)	GO:0016740	transferase activity
(1,8)	GO:0016301	kinase activity
(2,8)	GO:0006468	protein amino acid phosphorylation
Myotubularins		
(3,2)	GO:0006470	protein amino acid dephosphorylation
(4,2)	GO:0016787	hydrolase activity
(5,2)	GO:0016787	hydrolase activity
(5,3)	GO:0016787	hydrolase activity
(3,4)	GO:0016787	hydrolase activity
Collagens		
(6,1)	GO:0005581	collagen
(5,5)	GO:0005581	collagen
(7,5)	GO:0005201	extracellular matrix structural constituent
(1,6)	GO:0030198	extracellular matrix organization
(4,6)	GO:0007155	cell adhesion
(6,6)	GO:0007155	cell adhesion
(3,7)	GO:0007155	cell adhesion
(5,7)	GO:0007155	cell adhesion
(8,7)	GO:0005201	extracellular matrix structural constituent
(4,8)	GO:0007155	cell adhesion
(5,8)	GO:0007155	cell adhesion
(7,8)	GO:0005581	collagen

Table 5.2: Two Gaussian Clouds Properties

No. of points	Mean μ	Covariance Σ
50	(0,10)	$4 \cdot I$
50	(0,0)	$16 \cdot I$

5.1.5 Results

Two Gaussian Clouds

This example illustrates how the OSOM and Kohonen's SOM are equivalent for the $s^{(GOP)}$ similarity measure and the standard prototype update, Eq. (5.8). The dots in Fig. 5.7 show the object-data $\vec{T}$ for this example—two 2-dimensional Gaussian-distributed clouds. The object-data represents the terms. Table 5.2 outlines the properties of each cloud of data. The pair-wise term similarity matrix is

$$R_{ij} = 1 - \frac{||\vec{x}_i - \vec{x}_j||}{\arg\max_{kl} ||\vec{x}_k - \vec{x}_l||}.$$

Assume that the training data is composed of randomly-drawn normalized binary-combinations of the data in each cloud, much like the gene products are combinations of terms. The circles in Fig. 5.7 represent the 60 training data-points, i.e. the genes or gene products. Both the OSOM and the SOM are initialized to the same starting points with a 10×10 toroidal grid. Identical initialization is accomplished by randomly initializing the OSOM weight vectors, $\vec{w}_i \in [0,1]^2$, and then computing the corresponding weights of the SOM prototypes,

$$\vec{a}_i = \sum_{j=1}^{100} (\tilde{w}_i)_j \vec{T}_i.$$

Each algorithm was run for 1,000 iterations with equivalent learning-rates and neighborhood functions.

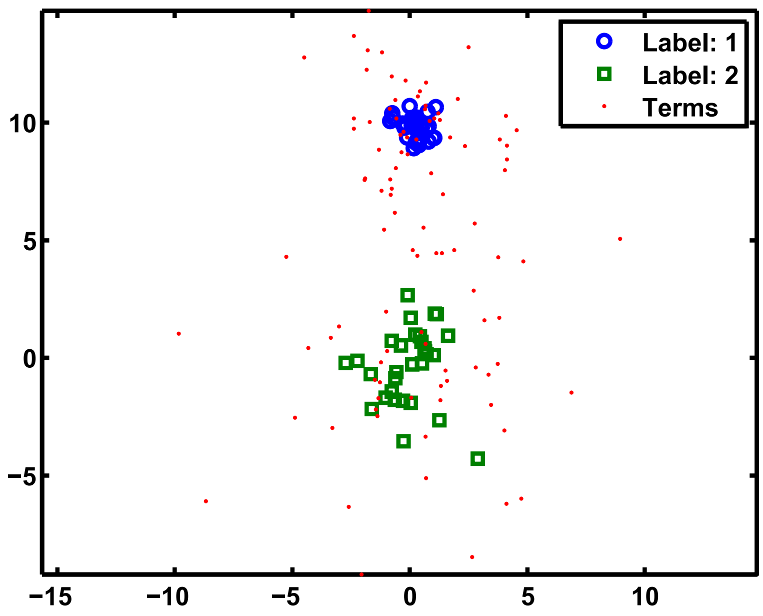
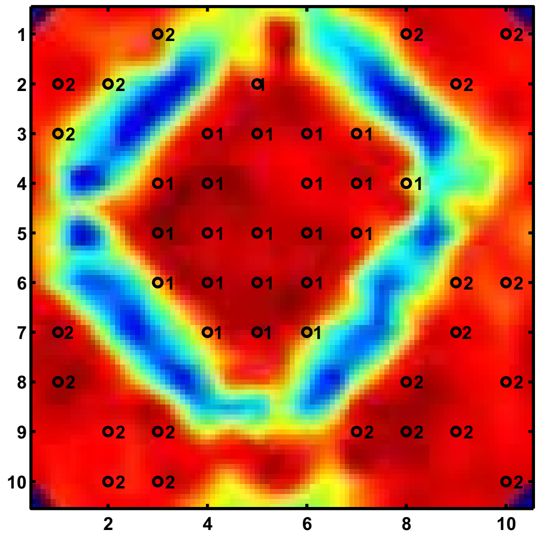
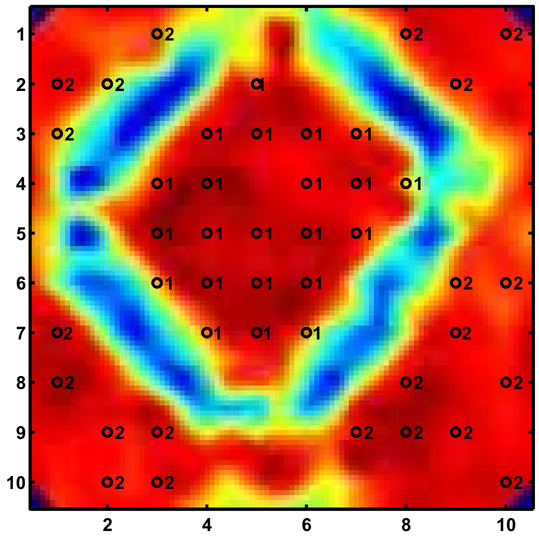


Figure 5.7: Two Gaussian Clouds



(a) OSOM



(b) SOM

Figure 5.8: Comparison of OSOM and SOM trained on two Gaussian clouds data shows identical results.

Figure 5.8 shows the comparison of the OSOM and SOM mappings of the two Guassian clouds data. The results are identical.

GPD194 Gene Products

To show the strength of the similarity-measure based OSOM, I compared against the SOSM [Ritter and Kohonen, 1989] and the batch relational SOM [Hasenfuss and Hammer, 2007]. As in [Ritter and Kohonen, 1989], I use a dot-product similarity measure for the SOSM,

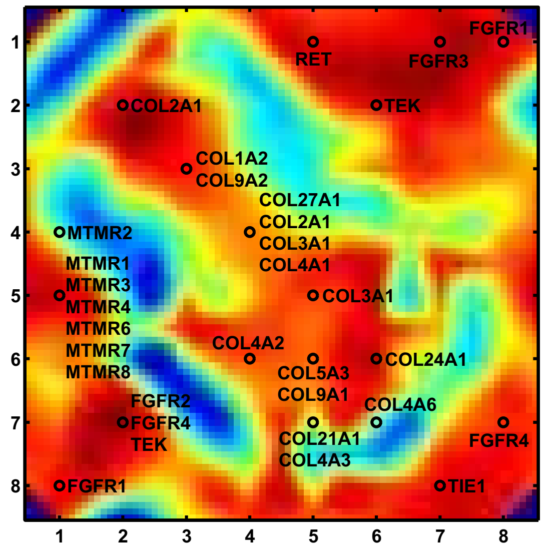
$$s^{(SOSM)}(\vec{w}_i, \vec{g}_j) = \vec{w}_i \cdot \vec{g}_j. \tag{5.13}$$

The SOSM update equation is equivalent to Eq.(5.8),

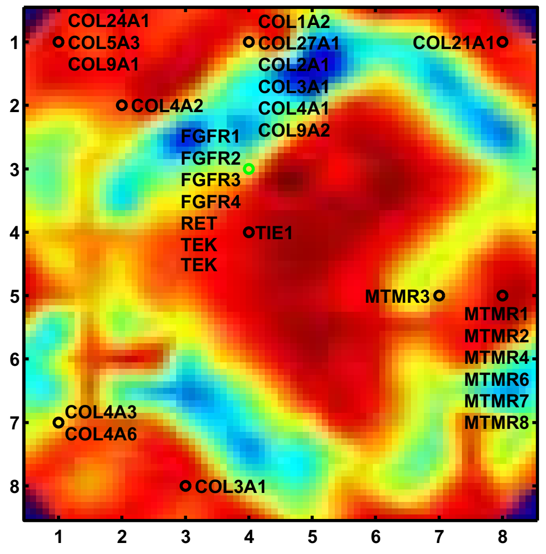
$$\vec{w}_i^{(new)} = \vec{w}_i^{(old)} + \epsilon(t) \cdot h_{ip} \cdot \left(\vec{g}_d - \vec{w}_i^{(old)}\right), \ \ \forall i. \tag{5.14}$$

The batch relational SOM trains on objects represented by dissimilarity data. However, this algorithm is unable to directly encode the ontological data as a binary vector of terms; thus, a pair-wise training-object dissimilarity matrix must be computed. I computed the dissimilarity matrix by $D_{ij}^{genes} = d^{(GOP)}(G_i, G_j), \forall i, j$. The batch relational SOM trains in a similar fashion to relational-duals of c-means clustering algorithms [Hathaway et al., 1989].

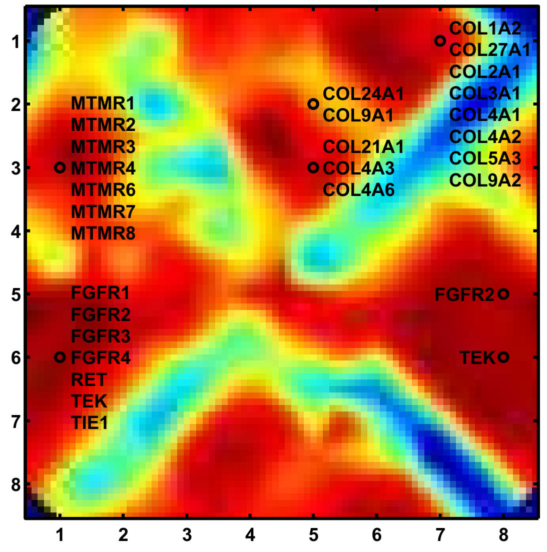
Figure 5.9 shows the visualizations of the trained OSOM networks using the $F^{(MAX)}$ prototype update equation for the three different similarity measures. All three visualizations show that the OSOM is able to correctly show the groupings of the three gene product families, the *collagens*, *receptor-precursors*, and *myotubularins* (the three families are described in Table 2.4. Interestingly, the OWA and AVG similarity measures produced smaller (but more populated) groups of gene products from each family. All three visualizations show that the OSOM mapped the *myotubu-*



(a) OSOM - $s^{(GOP)}$

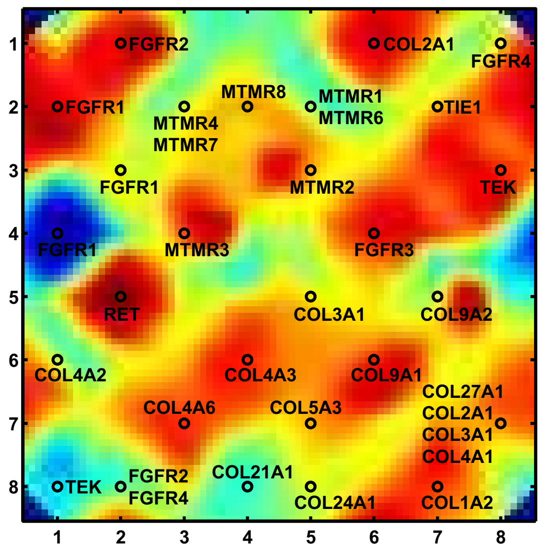


(b) OSOM - $s^{(AVG)}$

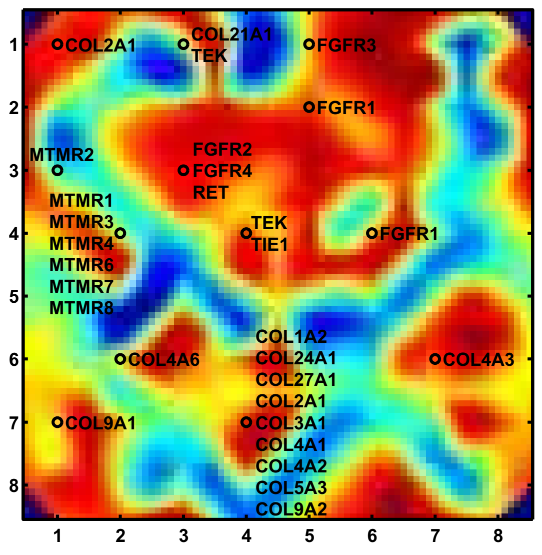


(c) OSOM - $s^{(OWA)}$

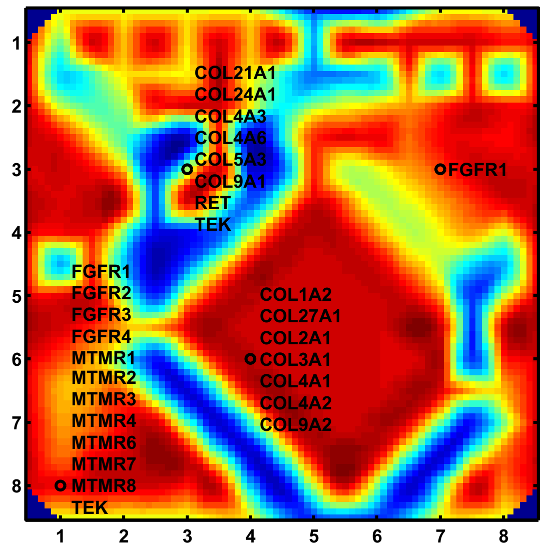
Figure 5.9: OSOM network mapping of $GPD194$ using $F^{(MAX)}$ update, eq.(5.11).



(a) Batch OSOM - $s^{(GOP)}$



(b) Batch OSOM - $s^{(AVG)}$



(c) Batch OSOM - $s^{(OWA)}$

Figure 5.10: Batch OSOM network mapping of $GPD194$.

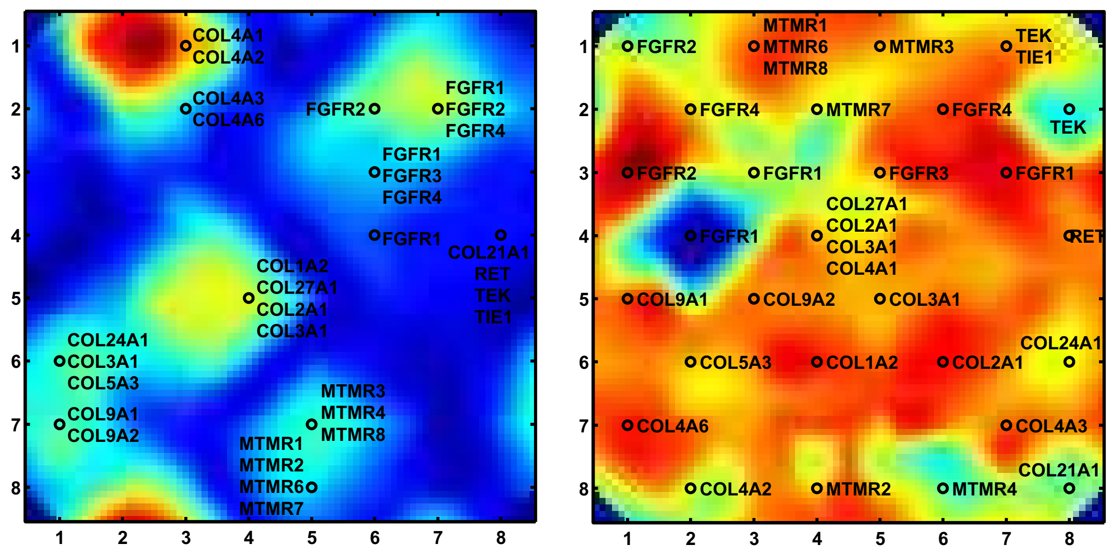
larins (MTMR) family onto a tightly grouped region. This family is annotated by nearly identical GO terms; hence, this follows my expectation. I also expected that the $F^{(MAX)}$ update would tend to group objects together as the update is moving the prototypes towards all terms that are similar to the terms in the test signal. This produces less specific prototypes.

Figure 5.10 shows the visualizations of the trained batch OSOM networks for the three different similarity measures. In contrast to the $F^{(MAX)}$ networks shown in Fig. 5.9, the batch OSOM produces a more spread out mapping of the gene products (with the exception of the OWA similarity). This is because the update only moves the prototypes toward the terms present in the gene products. Thus, the prototypes are more specific. The batch OSOM using $s^{(GOP)}$, shown in Fig. 5.10(a), produces the most pleasing result from an informational standpoint. The families are grouped in connected regions on the map, but the color visualization shows that there is substructure within the families. For example, the *collagen* gene products are separated into three red islands; the islands are centered at (5,6), (7,8), and (7,5). This substructure has been identified in other work [Popescu et al., 2004] and is also corroborated by the experiments described in [Myllyharju and Kivirikko, 2004]. Another pleasing aspect of this map is that gene products mapped to the locations (4,8), (1,8) and (1,4) are known outliers. The FGFR1, TEK, and COL21A1 gene products mapped to those locations are known to have wrong or incomplete annotations that cause them to be erroneously grouped (note that these annotation errors have since been corrected, but I use these data for consistency and validation purposes).

Figure 5.11 shows the results of the SOSM and batch relational SOM on the $GPD194$ data. The SOSM visualization, Fig. 5.11(a) is perhaps pleasing from the standpoint that it clearly shows the delineation between the three families, including the substructure in the *collagens* and the known outliers at locations (8,4) and (7,4).

However, the SOSM, in Fig. 11(a), fails to capture the underlying structure of the similarity between the three families. This family separation in the SOSM is caused because the SOSM does not consider the term-based similarity, it only calculates the similarity by a dot-product. The three families do not share any terms between them; thus, the locations on the SOSM map are very separated. The families do, however, share similar terms and this similarity is not captured in the SOSM.

The results of the batch relational SOM, shown in Fig. 5.11(b), are pleasing. The map is well-populated, the family structure is somewhat evident, and the underlying similarities between certain members of different families are shown. This map is very similar to the batch OSOM map shown in Fig. 5.10(a). This is expected. Each of these maps are computed using equivalent similarity measures, namely the GOP measure, and are updated using a batch algorithm. There are differences between the OSOM and batch relational SOM, though. First, the RSOM map does not show the boundaries between the families. It looks as though all prototypes are fairly similar to each other, except for the outliers at (8,8), (8,2) and (2,4). Also, the strength of the OSOM formulation is that the linguistic content of each prototype is directly encoded in the prototype weight vector—each weight vector element represents a term in the GO. Thus, the summarizations of the network locations are directly encoded in the map. In contrast, the batch relational SOM prototypes encode the weight of each object (in this case, gene product) in defining the map. Hence, each weight vector element is the contribution of a training object, rather than the terms themselves. Term-based summarizations could be computed from the batch relational SOM by aggregation or counting methods such as those described in [Popescu et al., 2004].



(a) SOSM (b) Batch Relational SOM

Figure 5.11: SOSM and batch relational SOM network mappings of $GPD194$.

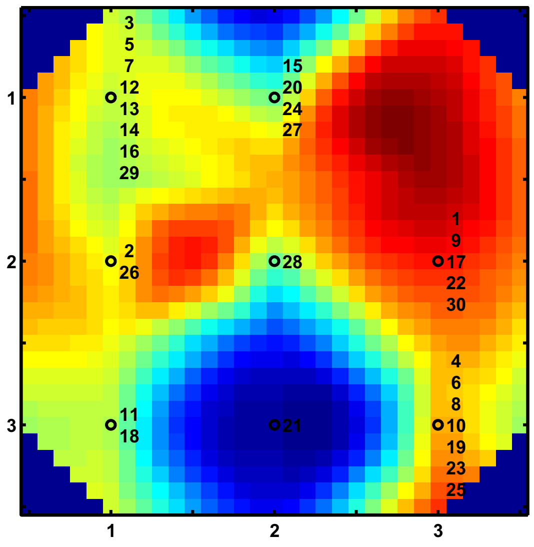
Cell-Apoptosis Related Genes

Table 5.3 outlines 30 genes that are known to be related to cell-apoptosis, or programmed cell death [Xu et al., 2008]. Genes 1-10 are known to be anti-apoptotic; these genes prevent cell death. Genes 11-19 are pro-apoptotic, they are involved in initiating cell death. Genes 20-30 are involved in apoptosis, but the GO annotations do not define them to be either pro- or anti-apoptotic. These genes are important to understanding the mechanisms of several cell-related diseases. Some cancer-causing viruses, including *human papilloma virus* (HPV), prevent cell-apoptosis in the compromised cells [Garnett and Duerksen-Hughes, 2006]. Other diseases, such as AIDS and Alzheimer's, cause cell-death [Cameron and Feuer, 2000]. Thus, it is important to understand the role(s) that genes play in cell-apoptosis.

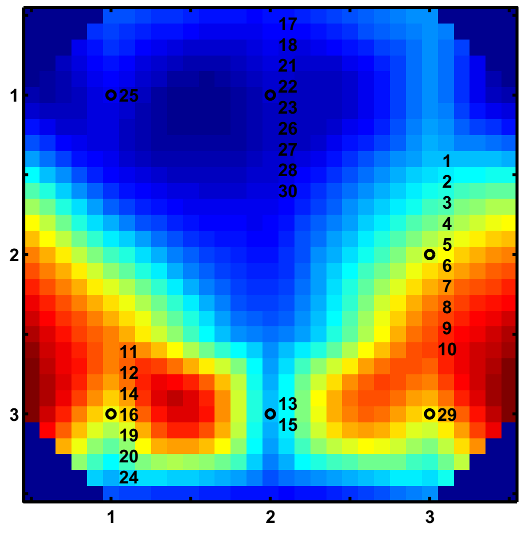
Figure 5.12 shows the results of training the batch OSOM, the SOSM, and the batch relational SOM with the cell-apoptosis genes in Table 5.3. Again, the SOSM shows the obvious result—all the anti-apoptosis genes are mapped to the same lo-

Table 5.3: Cell-Apoptosis Related Genes [Xu et al., 2008]

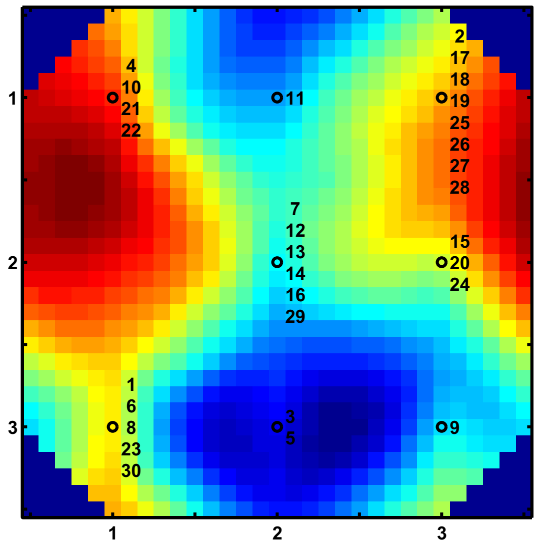
No.	Group	Gene names
1-10	Anti-apoptotic	RAF1, ANXA1, B2L10, BAG1, BCLW, BFL1, SOCS2, BNIP1, BCL2, COF1
11-19	Pro-apoptotic	ASC, BCL10, BCLF1, BIK, BIM, CASP3, CD2, P73L, TNFSF10, BAD
20-30	Involved in apoptosis	FOSL2, CD14, GAS2, CASP8, CD38, AHR, FAF1, P53, FAS, PAX3



(a) Batch OSOM - $s^{(GOP)}$



(b) SOSM



(c) Batch Relational SOM

Figure 5.12: Network mapping of cell-apoptosis-related genes.

cation, all the pro-apoptosis genes are mapped to two locations in the lower-left, and most of "unknown" genes (20-30) are mapped to the top-center. Notice that the "known" genes (1-19) are located on a red island that is connected, while most of "unknown" genes (20-30) are separated from the rest by a dark-blue boundary. As shown with the $GPD194$ data set, the SOSM captures the obvious relationships of genes, but it does not show any of the underlying similarities. Additionally, the functions of most of the "unknown" genes (20-30) cannot be inferred from this mapping.

In contrast, both the batch OSOM in Fig. 5.12(a) and the batch relational SOM in Fig. 5.12(c) show some interesting mixing of the genes. Notice that several of the anti-apoptotic genes (4,6,8,10) are mapped to location (3,3) in the OSOM map. However, one pro-apoptotic gene (19) and two of the "unknown" set (23,25) are mapped to this location also. Although the MRT for this location is GO:0006916, anti-apoptosis, this term is not shared by genes 19, 23, and 25; the other term-based similarities map these genes into this location. I found that a couple of these specific associations shown by the batch OSOM are supported by the biomedical literature. BAG1 (4) and GAS2 (23) are involved in cocaine-induced changes in fetuses [Novikova et al., 2005] and BAG1 (4) and BAD (19) have been associated in survival of neuronal cells [Gtz et al., 2005]. Another interesting map location in the batch OSOM is (1,1). There are three anti-apoptotic genes (3,5,7), four pro-apoptotic genes (12-14,16), and one of the unknown set (29). All of these genes share the term GO:0005515, protein binding, as well as other protein-binding terms. These types of relationships are important for biologists who wish to determine the links between genes which may not share the same annotations, but do share similar ones (as measured by the ontology-based similarity measures).

The batch relational SOM mapping of the cell-apoptosis related genes is effective in showing the relationships between the genes and is very similar in appearance to

Table 5.4: Most Representative Terms (MRT) of OSOM Map of Cell-Apoptosis Related Genes

OSOM Index	Gene Indexes	GO ID	GO Definition
(1,1)	3, 5, 7, 12-14, 16, 29	GO:0005515	protein binding
(1,2)	2, 26	GO:0006629	lipid metabolic process
(1,3)	11, 18	GO:0006917	induction of apoptosis
(2,1)	15, 20, 24, 27	GO:0005515	protein binding
(2,2)	28	GO:0030154	cell differentiation
(2,3)	21	GO:0008219	cell death
(3,2)	1, 9, 17, 22, 30	GO:0006915	apoptosis
(3,3)	4, 6, 8, 10, 19, 23, 25	GO:0006916	anti-apoptosis

the OSOM. Upon further inspection, there a similar groupings in the batch relational SOM and OSOM maps. However, recall that the batch relational SOM does not directly encode the term-content in the network prototypes. Thus, cluster summarization is not straight-forward, as it is with the OSOM. One must also go through the additional step of computing the pair-wise gene similarity matrix, upon which the batch relational SOM operates.

Table 5.4 contains the MRTs for each map location in the batch OSOM map of the cell-apoptosis related genes. Two of the groups have an MRT that suggests anti-apoptotic or pro-apoptotic related function, OSOM locations (1,3) and (3,3). The locations (2,3) and (3,2) have an MRT that is apoptosis-related but these apoptosis annotations are specific and only suggest that the genes are apoptosis-related.

Table 5.5 provides summarizing remarks about the variants of the OSOM. Overall, I found the most pleasing results to be the batch OSOM with the $s^{(GOP)}$ similarity measure both for the quality of the visualization as well as the linguistic summarizations.

Table 5.5: Summary of algorithms' behaviors

Similarity	Algorithm	Behavior
$s^{(OWA)}$	OSOM, $F^{(MAX)}$	Only considers top term-matches when computing similarity. Displays course grouping structure of the objects.
$s^{(AVG)}$	OSOM, $F^{(MAX)}$	Displays more substructure than the OWA based similarity. Better for determining "second-order" grouping structure.
$s^{(GOP)}$	OSOM, $F^{(MAX)}$	Displays finer grouping structure of object. Although, not as effective as the Batch OSOM with $s^{(GOP)}$ at producing a map that has no "dead" prototypes.
$s^{(OWA)}$	Batch OSOM	Very similar behavior as the OSOM with the $F^{(MAX)}$ update with the $s^{(OWA)}$ measure.
$s^{(AVG)}$	Batch OSOM	Very similar behavior as the OSOM with the $F^{(MAX)}$ update with the $s^{(AVG)}$ measure.
$s^{(GOP)}$	Batch OSOM	The most visually pleasing results. There are very few "dead" prototypes (the map is filled nicely). The course grouping is obvious but the substructure of the relationships between the objects is evident.

5.1.6 Perspectives on OSOM

The results in Section 5.1.5 show that the OSOM is a powerful tool for visualizing the relationships between objects composed of ontological data. Because these data are represented as collections of terms, the standard SOM is ill-equipped for these data. The results of the SOSM show that it is able to show the obvious relationships between the genes and gene products. However, the substructure and, more importantly, the relationships between gene products that do not share identical terms are not shown.

Section 5.1.5 illustrated that the OSOM with the GOP similarity measure is equivalent to the SOM for object data. However, the SOM cannot take ontological data as input. Thus, the OSOM can do everything the SOM is able to, but it also can analyze ontological data. The OSOM encodes the ontological data directly and computes a visualization of the gene products that shows how they are related to one another. Additionally, the weight values in the OSOM prototypes are the relative strength of each GO term in defining the genes and gene products mapped to that prototype. Each prototype is essentially a sentence which describes the mapped genes and/or gene products.

Similar to the OSOM, the batch relational SOM provided accurate and meaningful visualizations of the genes and gene products. However, this is achieved at an additional cost: the pair-wise gene dissimilarity matrix must be precomputed. Additionally, it is not straight-forward to compute a term-based summarization of each prototype.

The drawback to the OSOM is that it requires the ontology-based objects to be described as vectors, where each element represents a specific term. The Gene Ontology, as of 2009, has approximately 30,000 terms. Thus, if the OSOM was used with the entire Gene Ontology, each gene or gene product vector would have 30,000 elements, with only a few being non-zero. For this case, the OSOM would be computationally expensive. To combat this drawback, algorithms could be developed that operate within the ontology tree itself.

Chapter 6

Perspectives and Open Problems

The analysis and algorithms presented in this dissertation elucidate the natural groupings of objects described by relational data.

The main contributions of this dissertation are summarized as:

1. The notion of the *aligned partition* was defined and it was shown the number of possible aligned partitions is significantly smaller than the number of possible partitions, for a given data set.

2. It was proven that there is a direct relationship between the VAT algorithm, SL clustering, and Dunn's cluster validity index.

3. A recursive formulation of the iVAT algorithm was developed that produces both the VAT and iVAT images in $O(n^2)$ operations, as opposed to the original iVAT formulation that only produces the iVAT image in $O(n^3)$ operations.

4. A clustering algorithm was developed, called CLODD, which computes aligned partitions of (VAT)-reordered dissimilarity matrices;

5. A new formulation of the co-VAT algorithm was developed that significantly

improves the effectiveness of co-VAT in showing the cluster tendency of rectangular data.

6. The iVAT distance transformation was extended to co-VAT, resulting in the co-iVAT algorithm, which was shown to be (usually) more effective that co-VAT.

7. CLODD was extended to a special kind of relational data called rectangular data. The resulting algorithm, ReSL, computes 4 types of partitions of rectangular data and a degree of co-clusterness matrix, which describes the co-clusterness properties of the data set.

8. Lastly, the SOM was extended to ontologies. The OSOM algorithm produces SOM-type maps of data represented by ontologies and linguistic "cluster" summarizations of the resulting map.

Aligned partitions, VAT, and iVAT

Aligned partitions mimic the blocky nature of VAT images. I showed that for n objects, the ratio of the number of possible aligned partitions to the number of possible partitions is $\frac{|\mathbf{M}^*_{hcn}|}{|\mathbf{M}_{hcn}|} \approx \frac{n^{c-1}}{c^{n-1}}$, $c << n$. Applying this ratio for the fairly typical problem of $c = 10$ and $n = 10,000$ yields $|\mathbf{M}^*_{hcn}|/|\mathbf{M}_{hcn}| \approx 1/10^{9963}$ — a *very* small number. Thus, looking for aligned partitions of data is a much more tractable problem.

The blocky nature of VAT images is also related directly to SL and Dunn's validity index. I showed that SL clusters will *always* manifest as aligned partitions of VAT-reordered dissimilarity data. I leveraged this to prove that the contrast of the on-diagonal blocks and off-diagonal areas of the VAT images is exactly Dunn's index. Thus, Dunn's index, for a given SL c-partition, is a measure of the VAT image in showing a cluster tendecy of c.

An improved version of VAT, iVAT, is shown to produce more pleasing results for "tough" cases, such as the Three Lines example. However, the original iVAT formulation was very computational expensive. I developed and proved a recursive formulation of iVAT that produces not only the iVAT image, but also the VAT image, in an order-of-magnitude less operations than the original iVAT formulation. The computational complexity of my iVAT formulation is also equivalent to the original VAT algorithm. It is my opinion that iVAT will always perform at least a good, if not better, than VAT. Thus, my recursive formulation, which produces both VAT and iVAT images, is a significant advance over both the original VAT and iVAT algorithms.

A question that remains is how the iVAT image relates to SL clusters. Because VAT and iVAT share the same ordering, the Propositions in Section 3.2 hold true for iVAT also. However, the relationship between Dunn's validity index and iVAT is unexplored. It is my conjecture that iVAT shows the preferred SL partition(s) for a given data set. This conjecture is based on the fact that iVAT is able to accurately show the visually preferred cluster tendency of the Three Lines example. Also, I showed that the distances in the iVAT matrix are *all* MST edge weights. Hence, because of the direct relationship between SL clustering and the formation of the MST, I further believe that there is an intimate relationship between iVAT and SL—perhaps iVAT is more closely related to SL than VAT itself.

CLODD

Because of the relation between aligned partitions, VAT, and SL, I examined a method for computing aligned partitions from VAT-reordered dissimilarity data. CLODD uses image processing methods to compute partitions from VAT images.

CLODD answers the three most important questions in clustering:

1. How many clusters are there?

2. What are the clusters?

3. How good is the partition?

It was shown that CLODD performs very well when VAT performs well. However, not surprisingly, if VAT fails, so does CLODD.

Unlike some relational clustering algorithms, CLODD does not require the dissimilarity data to be produced with a Euclidean relation and it can work with all reordered dissimilarity data, regardless of the reordering method.

In Chapter 3, I proved relationships between VAT and SL. Because VAT is the primary method for reordering dissimilarity data as a preprocessing step for CLODD, the question remains as to the relationship between SL and CLODD. I believe that it can be shown that if there exists a compact-separated partition (having a Dunn's index > 1) then CLODD will return this partition. However, my attempts to prove this have been unsuccessful to date.

co-VAT and co-iVAT

Rectangular dissimilarity data presents a unique challenge as almost all clustering algorithms, including CLODD, cannot be applied to this type of data. However, co-VAT is a version of VAT that is adapted to rectangular data. I developed a new formulation of co-VAT that is shown to be more effective at showing the co-clusterness in rectangular data. I also extended the iVAT algorithm to produce co-iVAT, which is an iVAT-based algorithm for rectangular data. Like my iVAT formulation, my co-iVAT algorithm produces both the co-VAT and co-iVAT images in the same order of computational complexity as the original co-VAT algorithm.

In the future, I will extend the analysis in Chapter 3 to rectangular data. For example, what does it mean to have compact-separated co-clusters? Or, can the principles of Dunn's index be extended to the partitions in rectangular data? Ultimately, I believe that the answer to these questions will have a direct relationship to co-VAT images, much the same as with VAT.

ReSL

Like CLODD, ReSL find clusters in relational data by first using a VAT-based reordering scheme. In contrast to CLODD, ReSL (aptly named) finds clusters in rectangular relational data. The dissimilarity data is first reordered using the co-VAT (or co-iVAT) algorithm. ReSL then produces a partition of the row objects, a partition of the column objects, a partition of the union of the row and column objects, and a partition of the rectangular data called co-clusters. In addition, ReSL computes a degree of co-clusterness matrix which is a measure of the co-clusterness of each of the candidate co-clusters. ReSL is unique as it is the only visual clustering algorithm to my knowledge that partitions rectangular relational data.

As a result of the examples presented in this dissertation, questions arose in regards to clustering in rectangular data. Perhaps the most important question is: What is a co-cluster? The examples of pure rectangular relational data presented in Chapter 4 challenged the rule-of-thumb conjecture that the number of co-clusters k_{co} is equal to the number of row clusters k_r plus the number of column clusters k_c minus the number of clusters in the union $k_{r \cup c}$. Answering the co-cluster question is important for understanding how to interpret the degree of co-clusterness matrix in ReSL. See Appendix B for a brief discussion on this topic.

OSOM

Lastly, I described a SOM-based algorithm that finds groups in ontological data. The OSOM was demonstrated on the GPD194 data set and was effective in finding the "expected" groups. The drawback to the OSOM data-structure is that it contains a vector element for each term in the ontology. Thus, for large ontologies (like the GO), this vector can be very large. This is perhaps an undesirable trait as users will probably only want 2 to 4 terms to summarize a group. The question that arises is then, can an SOM-based algorithm be created that operates entirely within the ontology? I suspect that the answer to this question is, yes.

To summarize, clustering in relational data is an important and significant problem. Many (if not most) types of data can only be represented as relations between objects. These data include documents, bioinformatics and medical informatics data, ontologies, etc. The larger body of clustering theory, which is based on numerical object data, cannot be applied to these data directly. This dissertation provides tools for examining these data visually and partitioning them into appropriate groups.

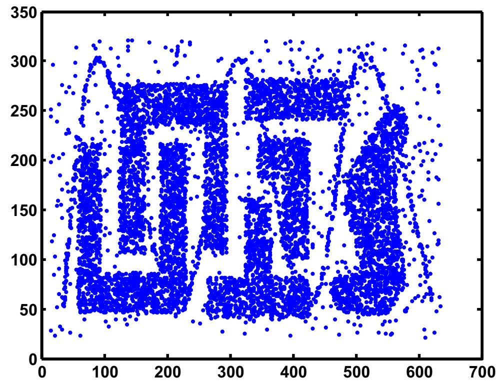
Appendix A

Additional Examples

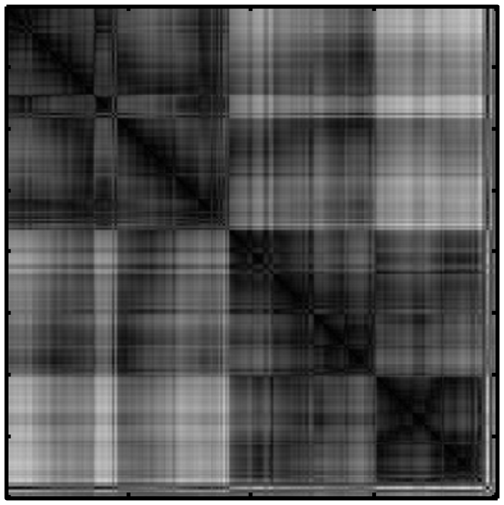
A.1 Recursive iVAT

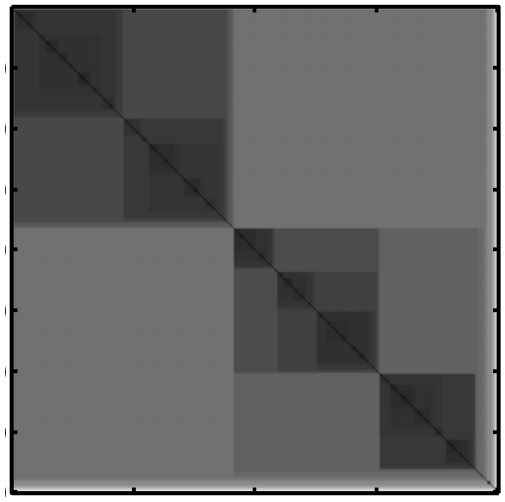
Figures A.1-A.3 show the VAT and iVAT images of sets of large-scale (on the order of 8,000 objects) noisy data. The data shown in these figures were originally created by Karypis et al. to demonstrate the CHAMELEON clustering algorithm. View (a) in each of these figures show the object data from which a dissimilarity matrix was computed by a Euclidean distance relation. Views (b) and (c) show the VAT and iVAT images, respectively. In view (d) of each of the figures, I picked, by eye, the dark blocks that I thought corresponded to the clusters in the data. The dark blocks I picked are shown by yellow dotted lines. Finally, view (e) shows the corresponding objects for each of the dark blocks I chose. By examining views (d) and (e), I can make a determination as to how effective iVAT is in showing the visually preferred clusters in these data.

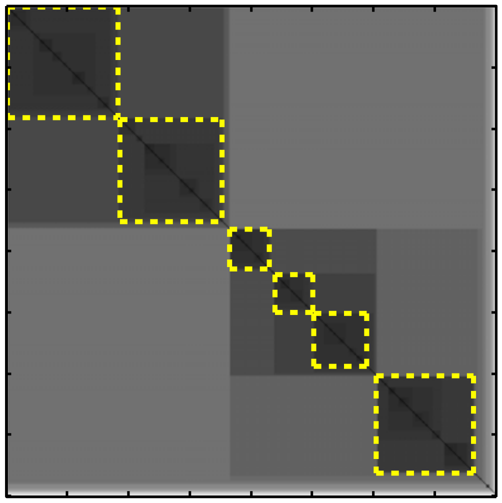
Let's examine Fig. A.1 first. View (a) suggests that there are 6 clusters in this data set — the 2 'U'-like groups on the left, the 3 groups in middle, and the 'S'-like structure on the right. The VAT image in view (b) does not clearly suggest any

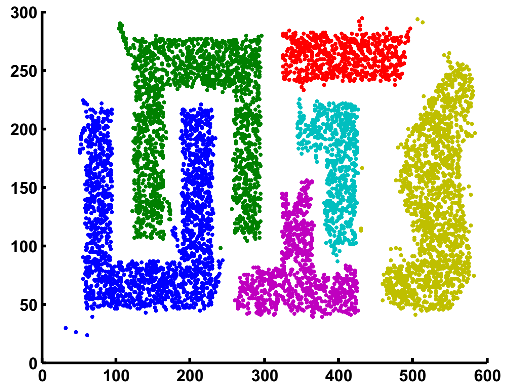


(a) Object data


(b) VAT image

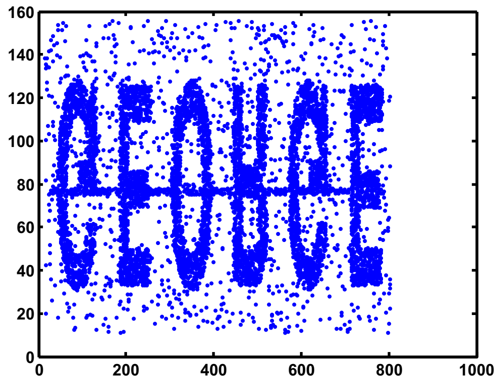

(c) iVAT image


(d) iVAT partition (by eye)

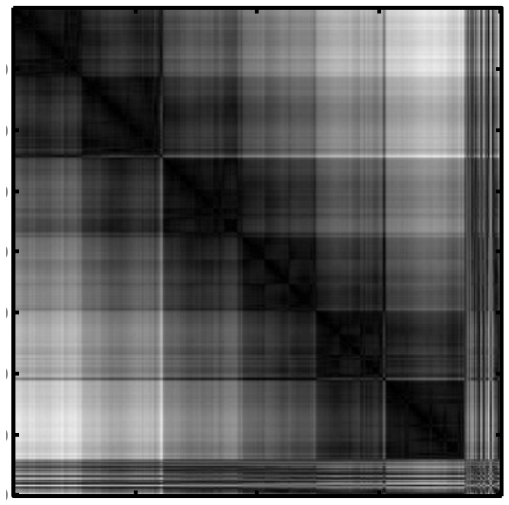


(e) Corresponding object data groups

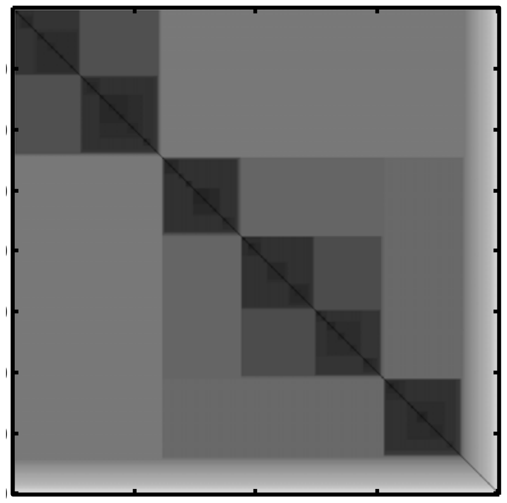
Figure A.1: Example of VAT and iVAT images in large-scale dissimilarity data composed of Euclidean relations on 8,000 objects [Karypis et al.].



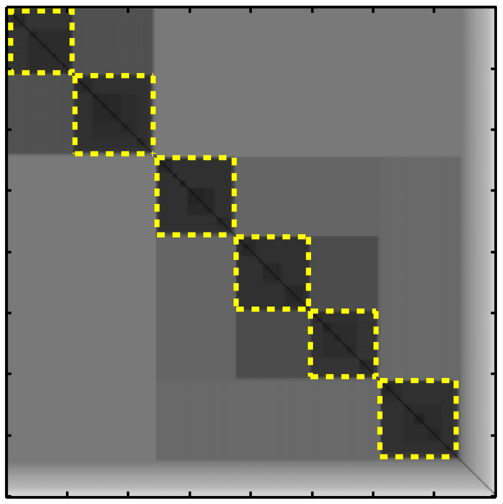
(a) Object data



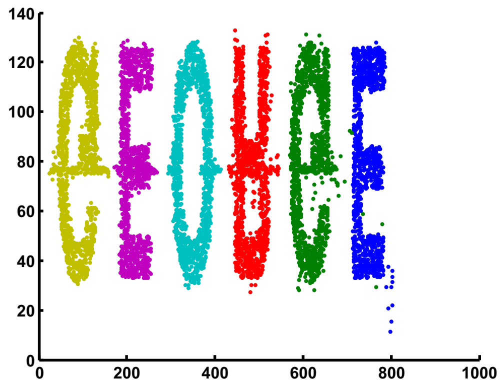
(b) VAT image



(c) iVAT image

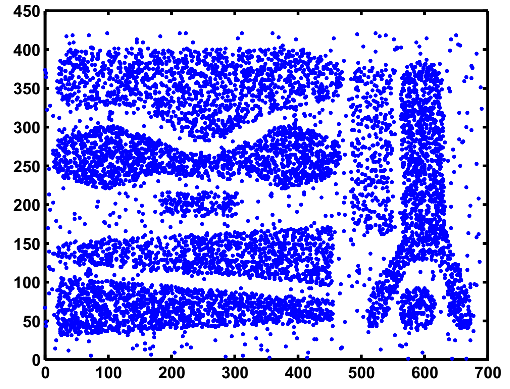


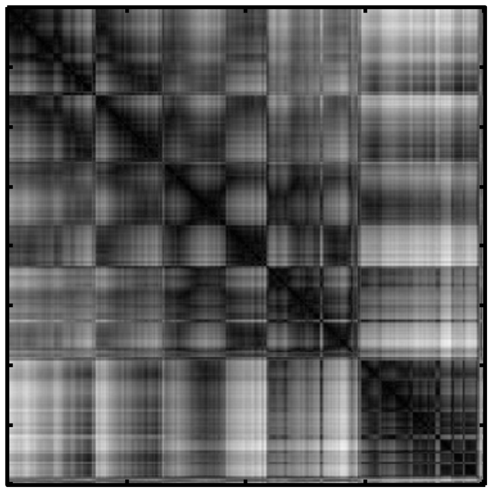
(d) iVAT partition (by eye)

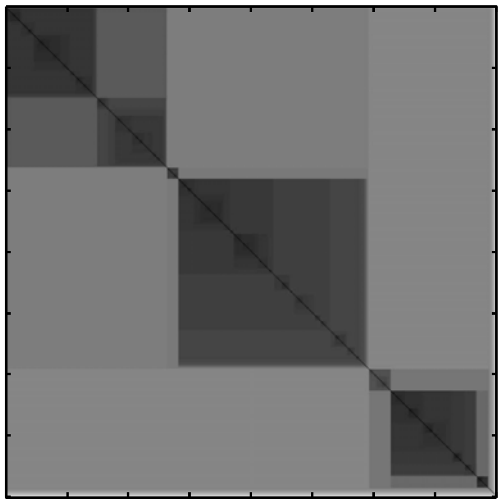


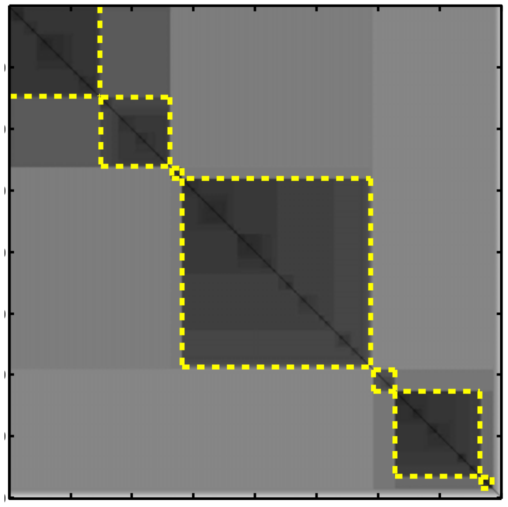
(e) Corresponding object data groups

Figure A.2: Example of VAT and iVAT images in large-scale dissimilarity data composed of Euclidean relations on 8,000 objects [Karypis et al.].


(a) Object data


(b) VAT image


(c) iVAT image


(d) iVAT partition (by eye)

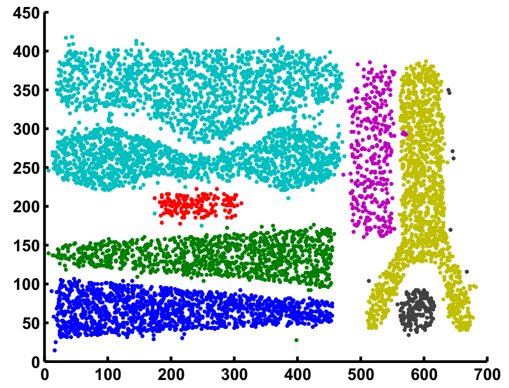

(e) Corresponding object data groups

Figure A.3: Example of VAT and iVAT images in large-scale dissimilarity data composed of Euclidean relations on 8,000 objects [Karypis et al.].

cluster tendency. However, the iVAT image in view (c) suggests to me that there are 6 clusters. I have outlined the dark blocks that I see in view (d). Finally, I plotted the corresponding objects in each dark block, one color per dark block, in view (e). Clearly, view (e) suggests that iVAT is successful in showing (to me) the 6 clusters in these object data.

The example shown in Fig. A.2 tell a similar story as the previous example. View (a) shows 8,000 objects; I suggest that there are six clusters in these data, one for each letter in "GEORGE". However, there is much noise in the data and, also, a line of objects that extends across "GEORGE" (one might imagine that this line could be a seventh cluster). The VAT image in view (b) somewhat shows 6 clusters by the 6 dark blocks. However, the iVAT image in view (c) presents a clear picture of 6 dark blocks. I have outlined these dark blocks in view (d) and view (e) shows the corresponding objects in each of these dark blocks, each plotted in a different color. Again, it is obvious that iVAT (and VAT as well, though not as obvious) is successful in showing the cluster tendency of these data.

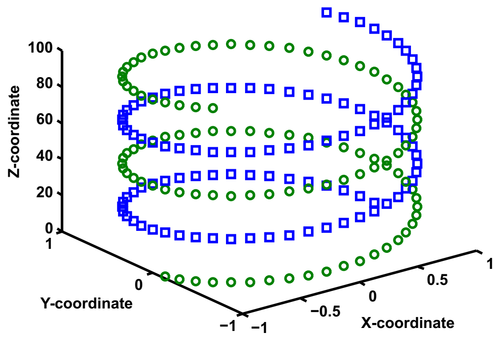
The last example, of this type, is shown in Fig. A.3. View (a) is a plot of 8,000 objects with (arguably) 8 groups — the 5 dense-regions on the left, the less-dense region at the horizontal coordinate 500, the upside-down 'Y'-like structure, and the small circle at the lower-right. The VAT image in view (b) does not suggest to me any clear cluster tendency. Again, however, the iVAT image makes the decision much more simple. I see 7 clusters in view (c), which I have outlined in view (d). View (e) plots the outlined groups from view (d), each in a different color. Unlike the previous two examples, iVAT is not completely successful in showing the cluster tendency. It "fails" to separate the two large clusters in the upper-right of the plot. This is because there are a small group of objects, formed in a line, around the horizontal coordinate 400 and vertical coordinate 300 that connects these two clusters. This line is so dense that it “fools” the path-based distance measure used in iVAT. iVAT is successful in showing the other clusters in these data.

A.2 CLODD

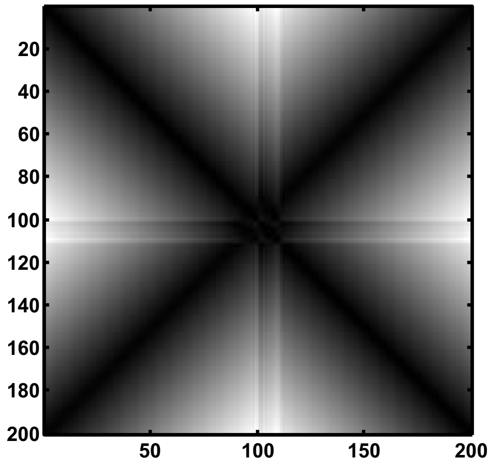
Figure A.4 demonstrates the flexibility of CLODD in using both VAT and iVAT images of dissimilarity data. View (a) shows the object data from which dissimilarity data was computed using a Euclidean distance relation. As this view shows, there are two groups of objects, each comprising a spiral strand (think DNA). As view (b) shows, VAT is unable to show the tendency to 2 clusters. However, the iVAT image, shown in view (c), clearly shows 2 clusters. Views (d) and (e) present the CLODD partitions of the VAT and iVAT images, respectively. Again, CLODD fails when VAT fails; however, CLODD is able to partition the data correctly when iVAT is used to produce the input dissimilarity image.

A.3 Alternate co-VAT Reordering

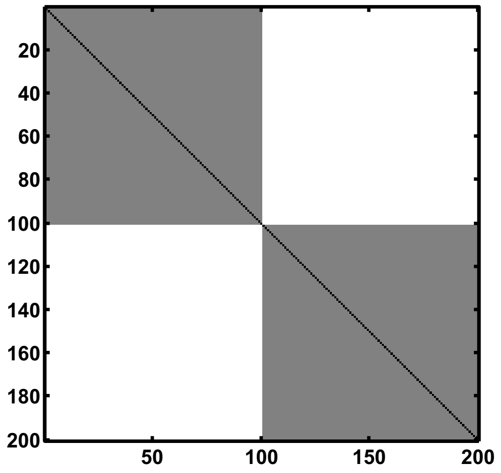
Figures A.5-A.7 show comparisons of the original co-VAT formulation, proposed in [Bezdek et al., 2007], to the new formulation, presented in Section 4.2. The datasets shown in these figures mimic selected examples from the original co-VAT manuscript [Bezdek et al., 2007]. Comparing the original co-VAT image in view (b) with the new formulation in view (c) shows that the new formulation is comparably effective in displaying the clustering tendency for these three examples. In the first two examples shown in Figs. A.5 and A.6, both the original and new formulations are successful in showing the clustering tendency. In the example shown in Fig. A.7, both formulations fail to show the correct tendency. This failure is discussed in [Bezdek et al., 2007]. In



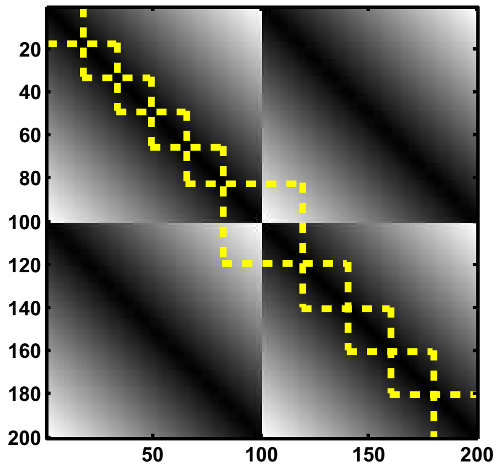
(a) Object data



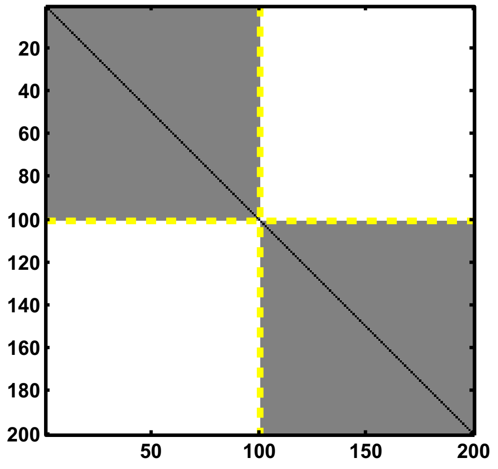
(b) VAT image



(c) iVAT image



(d) CLODD partition of VAT image



(e) CLODD partition of iVAT image

Figure A.4: CLODD partitions of objects organized in two spirals.

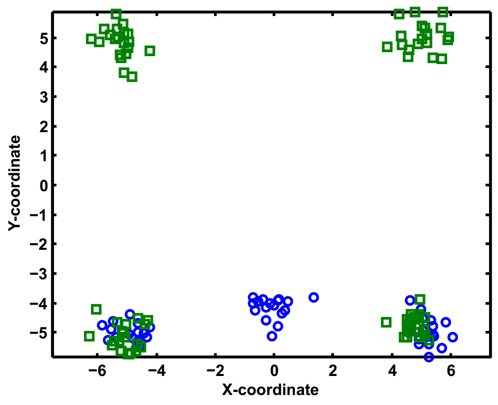
brief, because the two row clusters are approximately equidistant from each group of the column objects, the imputed distances in $\mathbf{D}_r$ and $\mathbf{D}_c$, as computed by Eqs.(2.22) and (2.23), are erred. These two dissimilarity matrices are important in the reordering of $\mathbf{D}$ in both the original and new formulations; thus, co-VAT fails for this data set.

Figure A.8 presents an example for which the alternate co-VAT reordering method is superior to the original reordering scheme. View (a) shows the 3-dimensional object data. There are 250 row objects organized in 5 line-like groups, shown as blue circles, and 125 column objects organized in a 'figure 8' pattern, shown as green squares. View (d) clearly shows that there are 5 clusters in the row objects, which is the visually preferred number of clusters. In contrast, view (e) does not show a clear dark block structure. Instead, there is a strong near-diagonal, with high dissimilarity shown in the off-diagonal. This type of structure is also shown in the Three Lines example, Example 3.2.2. It is my opinion that the image $\mathbf{D}_c^*$ shows that there is 1 cluster organized in a connected line pattern (which is supported by the plot of the object data). The image of $\mathbf{D}_{r\cup c}^*$, shown in view (f), clearly shows a tendency of 5 clusters. Thus, the rule of thumb would indicate $1 = 5 + 1 - 5$ co-clusters.

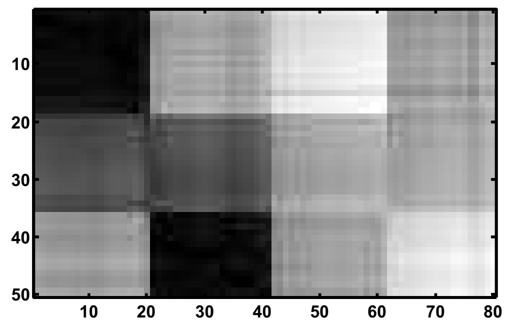
However, the alternate co-VAT image, in view (c), seems to indicate 5 co-clusters by the 5 dark blocks. However, the blocks are blurry. These 5 co-clusters are the 5 cross-like structures produced by the union of the row objects and column objects. In contrast, the co-VAT image in view (b) does not offer as clear a view of the co-cluster structure of these data.

A.4 co-iVAT

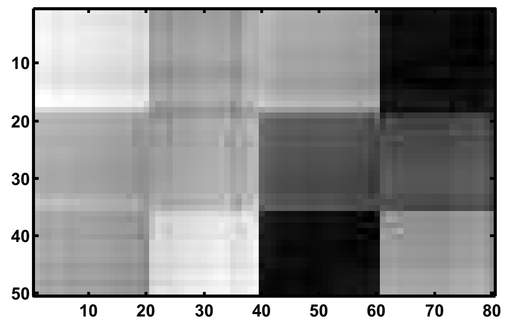
Figures A.9 and A.10 show the co-iVAT images for the examples presented in Figs. A.6 and A.7. The co-VAT algorithm was able to successfully show the cluster tendency



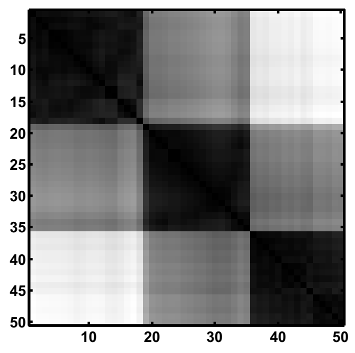
(a) Object data - (○) row objects, (□) column objects



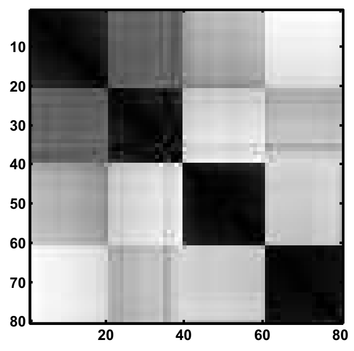
(b) $\mathbf{D}^*$



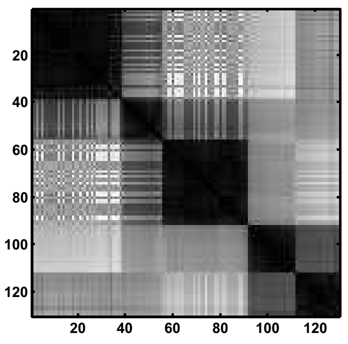
(c) alternate $\mathbf{D}^*$



(d) $\mathbf{D}^*_r$

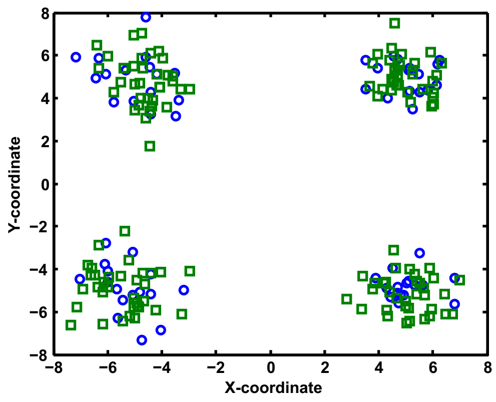


(e) $\mathbf{D}^*_c$

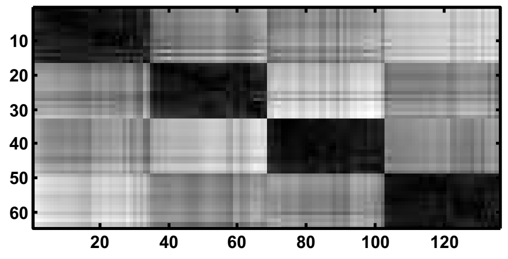


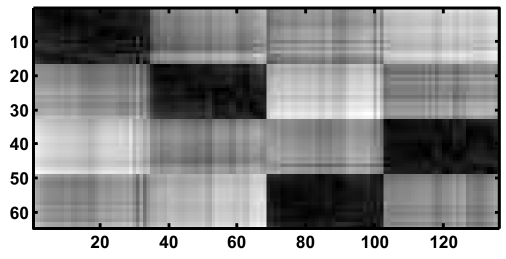
(f) $\mathbf{D}^*_{r\cup c}$

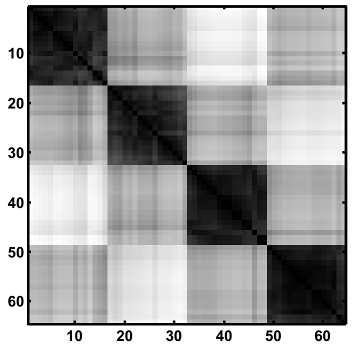
Figure A.5: Example that mimics the dataset shown in Fig. 1 from [Bezdek et al., 2007]. (a) - object data; (b) original co-VAT reordering, (c) alternate co-VAT reordering, (d-f) - co-VAT reordered dissimilarity data matrices.

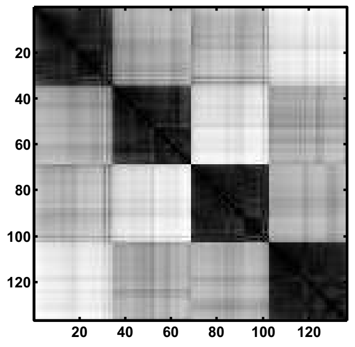


(a) Object data - (○) row objects, (□) column objects


(b) $\mathbf{D}^*$


(c) alternate $\mathbf{D}^*$


(d) $\mathbf{D}_r^*$


(e) $\mathbf{D}_c^*$

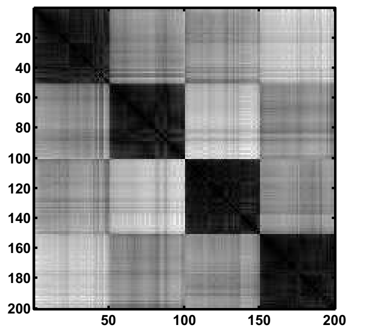
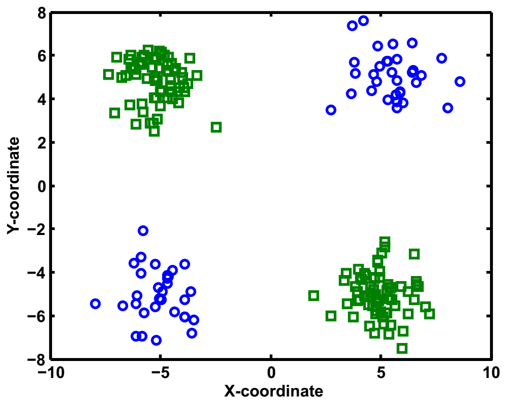
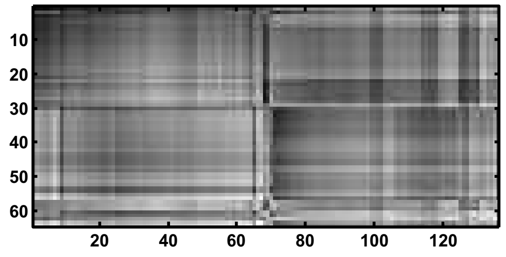

(f) $\mathbf{D}_{r\cup c}^*$

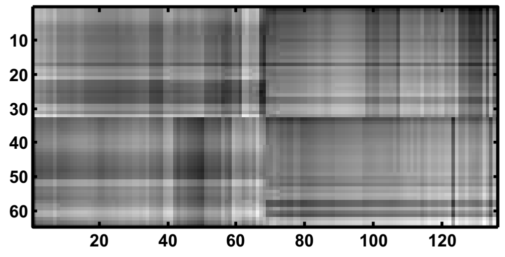
Figure A.6: Example that mimics the dataset shown in Fig. 4 in [Bezdek et al., 2007]. (a) - object data; (b) original co-VAT reordering, (c) alternate co-VAT reordering, (d-f) - co-VAT reordered dissimilarity data matrices.



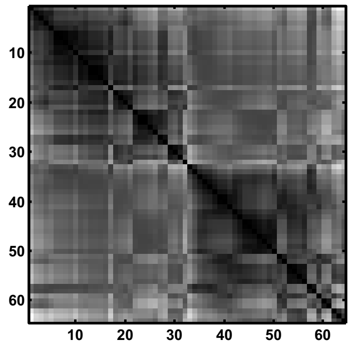
(a) Object data - (○) row objects, (□) column objects



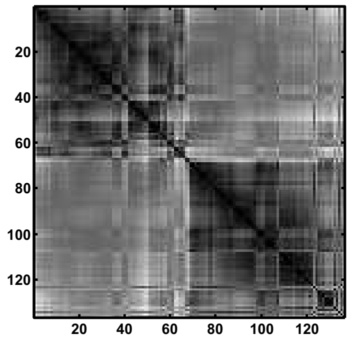
(b) $\mathbf{D}^*$



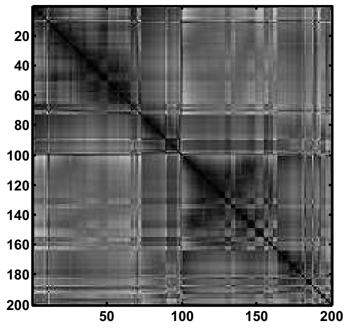
(c) alternate $\mathbf{D}^*$



(d) $\mathbf{D}^*_r$

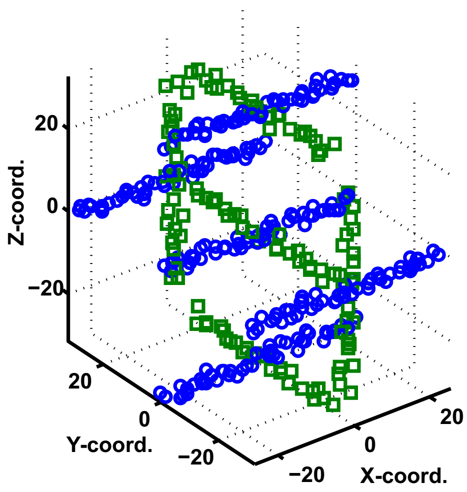


(e) $\mathbf{D}^*_c$

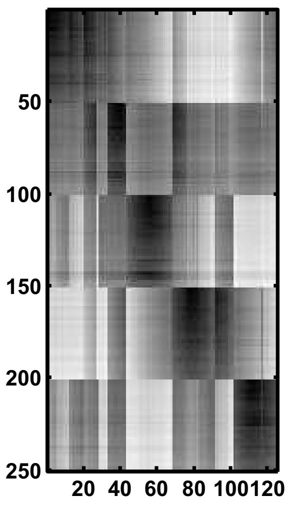


(f) $\mathbf{D}^*_{r \cup c}$

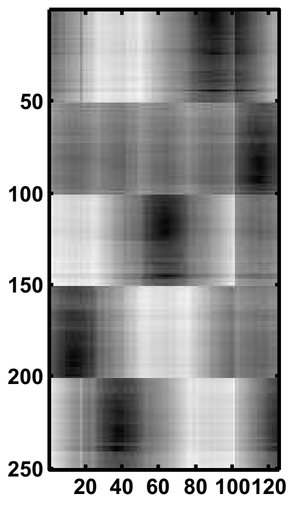
Figure A.7: Example that mimics Fig. 7 from [Bezdek et al., 2007]. (a) - object data; (b) original co-VAT reordering, (c) alternate co-VAT reordering, (d-f) - co-VAT reordered dissimilarity data matrices.



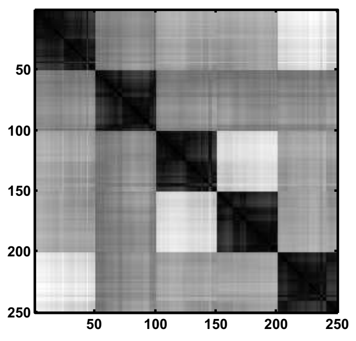
(a) Object data - (∘) row objects, (□) column objects



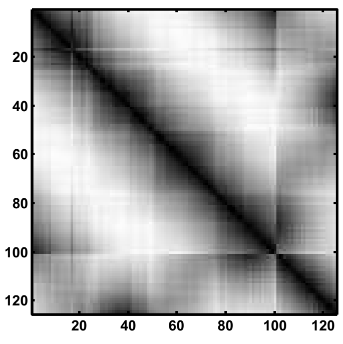
(b) $\mathbf{D}^*$



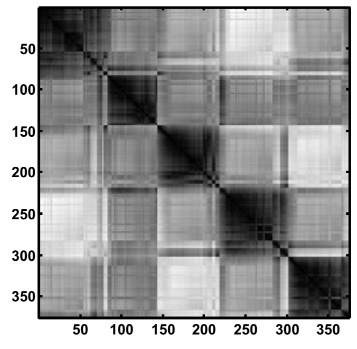
(c) alternate $\mathbf{D}^*$



(d) $\mathbf{D}^*_r$

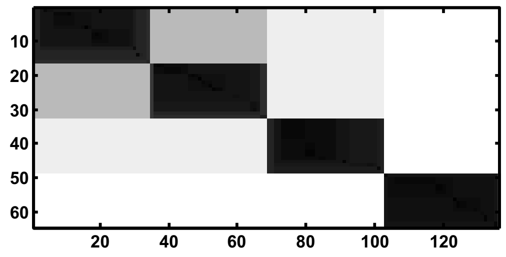


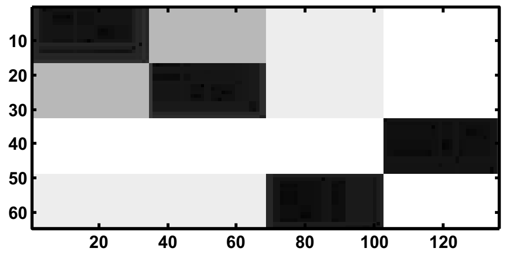
(e) $\mathbf{D}^*_c$

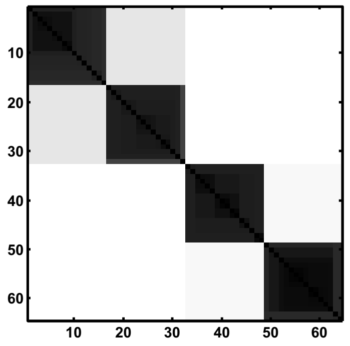


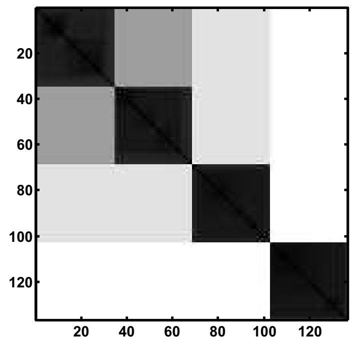
(f) $\mathbf{D}^*_{r\cup c}$

Figure A.8: Example of co-VAT algorithms on data created from 3-dimensional object data. (a) - object data; (b) original co-VAT reordering, (c) alternate co-VAT reordering, (d-f) - co-VAT reordered dissimilarity data matrices.


(a) $\mathbf{D}^*$


(b) alternate $\mathbf{D}^*$


(c) $\mathbf{D}^*_r$


(d) $\mathbf{D}^*_c$

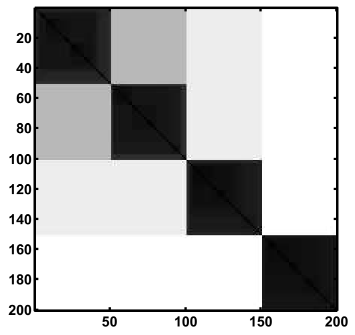
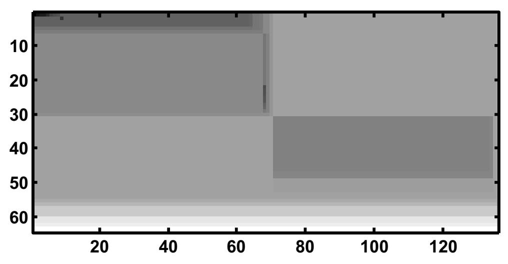
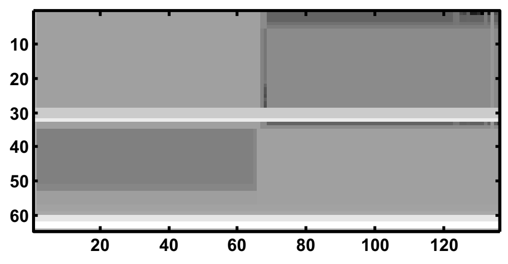
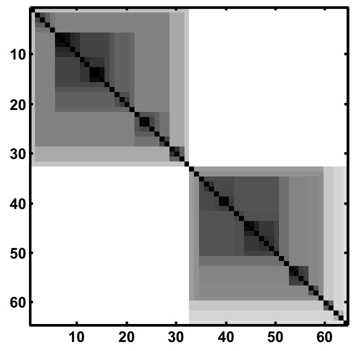

(e) $\mathbf{D}^*_{r\cup c}$

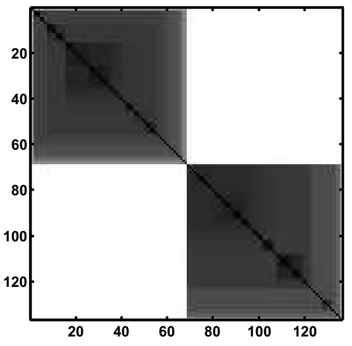
Figure A.9: co-iVAT images of dissimilarity data shown in Fig. A.6 (a) original co-iVAT reordering, (b) alternate co-iVAT reordering, (c-e) - co-iVAT reordered dissimilarity data matrices.

of the data shown in Fig. A.6(a) and co-iVAT shows similar effectiveness. However, co-iVAT was unable to show the cluster tendency of the data in Fig. A.7. The co-iVAT results, shown in Fig. A.10, are slightly more successful in showing the cluster tendency for this data. Views (c) and (d) show the cluster tendency in the row objects and column objects, respectively. Each of these views show the correct cluster tendency of 2 clusters. However, I think the reader will agree that the co-iVAT image of $D^*_{r\cup c}$, shown in view (d), fails to suggest a cluster tendency of 4 clusters in the union of the row and column objects. Because this data set does not have any true co-clusters it is unclear what the co-iVAT of $\mathbf{D}^*$ should show. In the previous example, shown in Fig. A.9, the co-clusters were clearly presented in the co-iVAT images of $\mathbf{D}^*$. In comparison, the images of $\mathbf{D}^*$ shown in Fig. A.10(a,b) shows no distinct dark blocks, which is supported by visual inspection of the object data in Fig. A.7(a).


(a) $\mathbf{D}^*$


(b) alternate $\mathbf{D}^*$


(c) $\mathbf{D}_r^*$


(d) $\mathbf{D}_c^*$

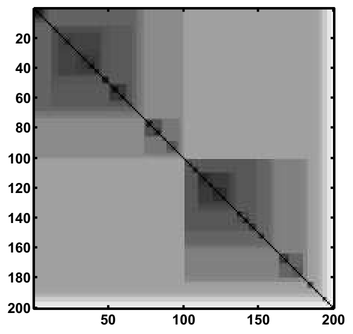
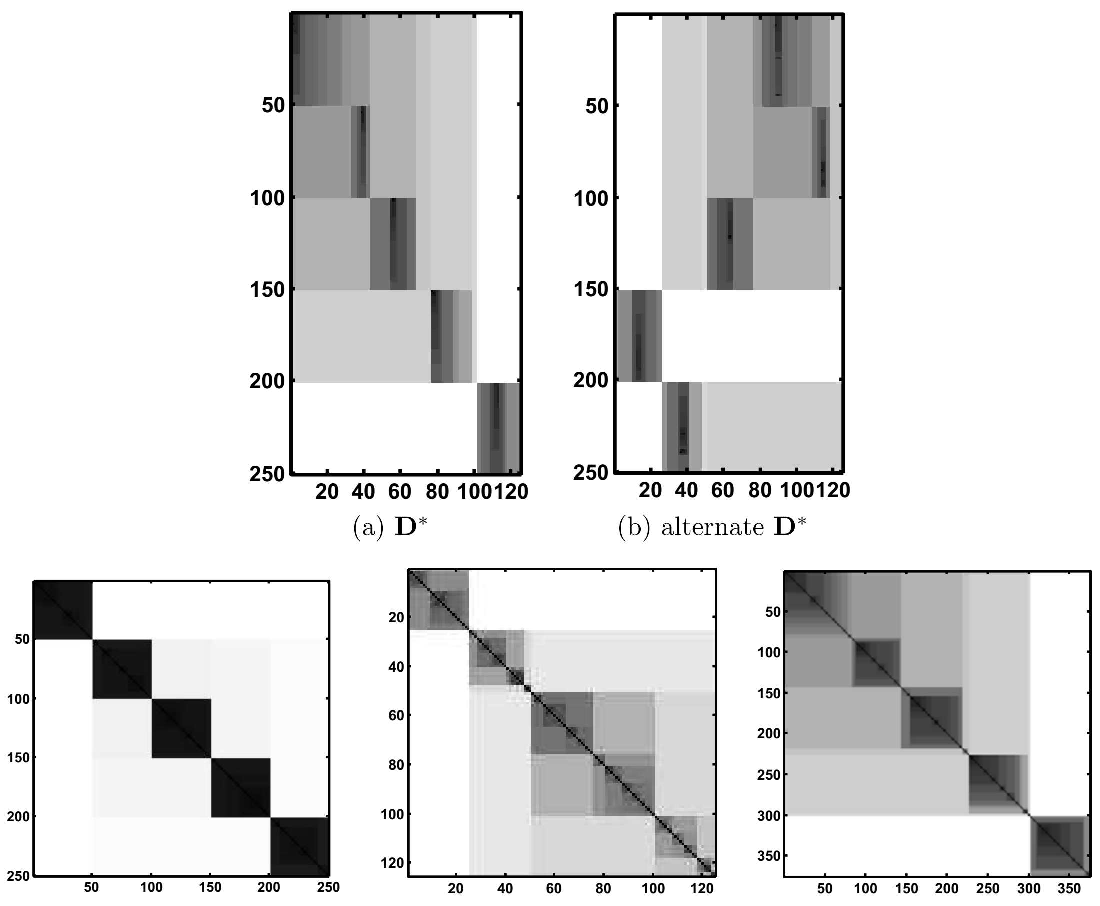

(e) $\mathbf{D}_{r\cup c}^*$

Figure A.10: co-iVAT images of dissimilarity data shown in Fig. A.7 (a) original co-iVAT reordering, (b) alternate co-iVAT reordering, (c-e) - co-iVAT reordered dissimilarity data matrices.

Figure A.11 shows the co-iVAT images of the dissimilarity data computed from the 3-dimensional data shown in Fig. A.8(a). As opposed to the co-VAT image shown in Fig. A.8(b), which was inferior to the alternate co-VAT image in (c), both co-iVAT images are successful at showing the 5 co-clusters in the object data. This leads me to believe that co-iVAT images derived from Euclidean relations will always be at least as good, if not better than, co-VAT images.

View (c) in Fig. A.11 clearly shows that there are 5 clusters in the row objects. Also, unlike the co-VAT image, $\mathbf{D}_c^*$, the co-iVAT image of the column object dissimilarity data, $\mathbf{D}_c'^*$, more closely resembles a VAT image which shows 5 clusters. View (e) shows 5 clusters in the union of the row and column objects, which is similar to the co-VAT image in Fig. A.8(f).

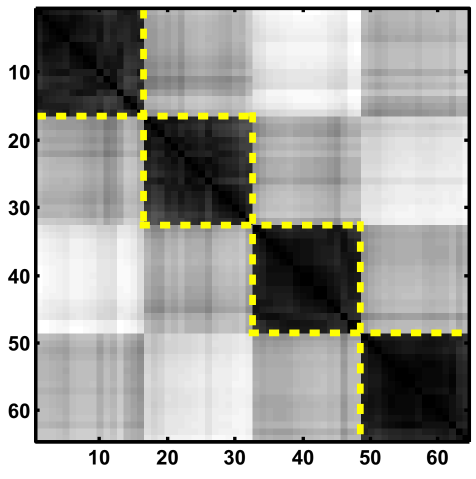

(a) $\mathbf{D}^*$ (b) alternate $\mathbf{D}^*$

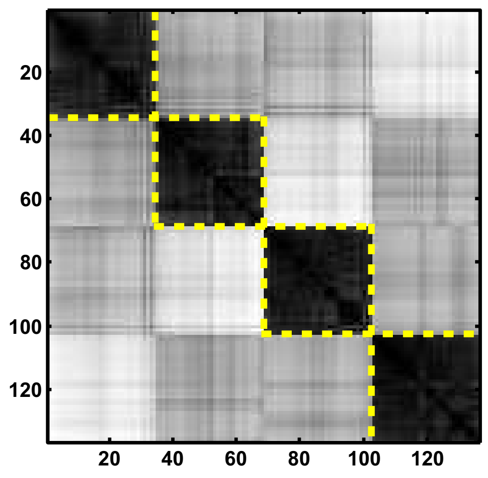
(c) $\mathbf{D}_r^*$ (d) $\mathbf{D}_c^*$ (e) $\mathbf{D}_{r\cup c}^*$

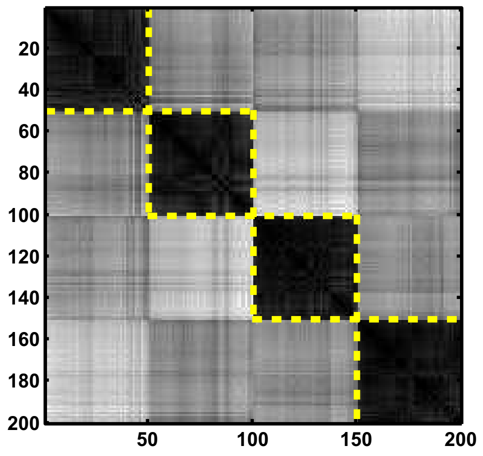
Figure A.11: co-iVAT images of dissimilarity data shown in Fig. A.8 (a) original co-iVAT reordering, (b) alternate co-iVAT reordering, (c-e) - co-iVAT reordered dissimilarity data matrices.

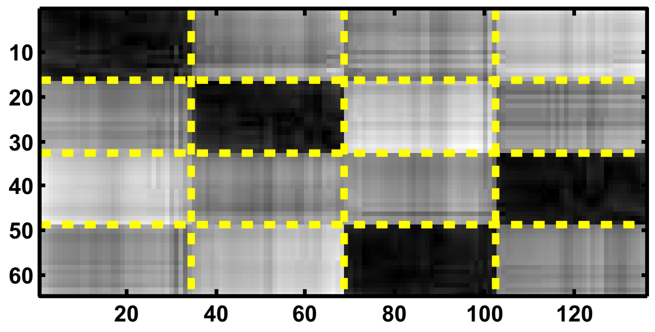
A.5 Rectangular Single Linkage

Figure A.12 shows the results of using ReSL on the co-VAT images in Fig. A.6. Again, the object for which $\mathbf{D}$ is computed from has 4 distinct clusters, each composed of both row and column objects. Views (a) and (b) show that there are 4 clusters in the row objects and 4 clusters in the column objects, respectively. View (c) shows that there are 4 clusters in the union of the objects. Finally, the view of the partitions outlined on the alternate co-VAT reordered dissimilarity data, shown in view (d), clearly shows that there are 4 co-clusters and which objects below to each of these


(a) Partition of $\mathbf{D}_r^*$


(b) Partition of $\mathbf{D}_c^*$

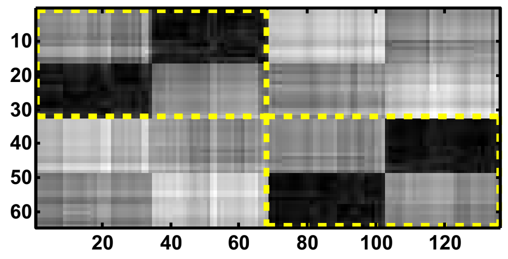

(c) Partition of $\mathbf{D}_{r\cup c}^*$


(d) Partition of $\mathbf{D}^*$

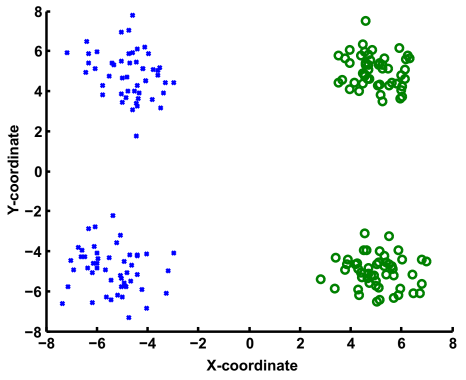
$$\begin{bmatrix} 0.87 & 0.32 & 0.28 & 0.01 \\ 0.31 & 0.87 & 0.02 & 0.31 \\ 0 & 0.33 & 0.29 & 0.90 \\ 0.27 & 0.06 & 0.90 & 0.32 \end{bmatrix}$$

(e) Partition values of $\mathbf{U}_{co}$

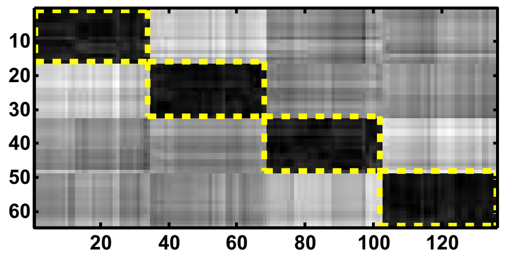
Figure A.12: ReSL partitions of co-VAT reordered dissimilarity images shown in Fig. A.6



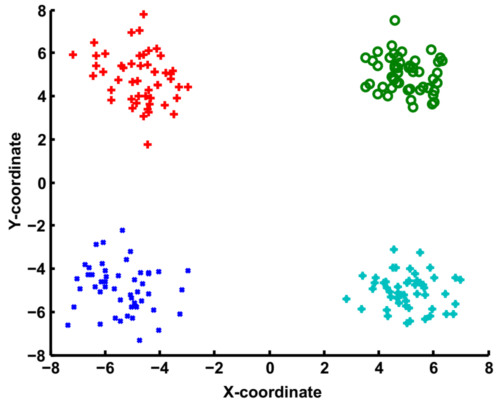
(a) 2-partition, dissimilarity data



(b) 2-partition, object data



(c) 4-partition, dissimilarity data



(d) 4-partition, object data

Figure A.13: Spectral co-clustering partitions of dissimilarity data shown in Figs. A.6 and A.12.

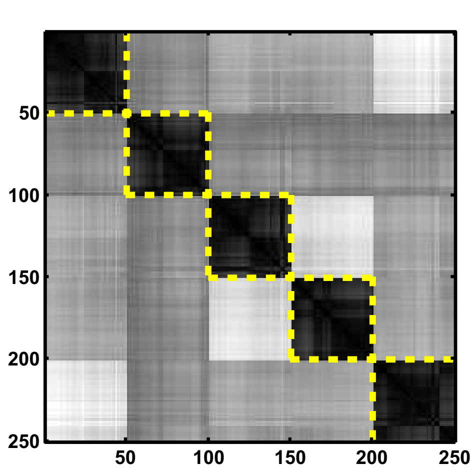
clusters. The degree of co-clusterness matrix in view (e) also supports this claim.

To compare, Fig. A.13 demonstrates the spectral co-clustering algorithm on the same data set. The views in this figure shows that spectral co-clustering is also successful in partitioning the data. However, recall that spectral co-clustering cannot automatically detect the number of co-clusters.

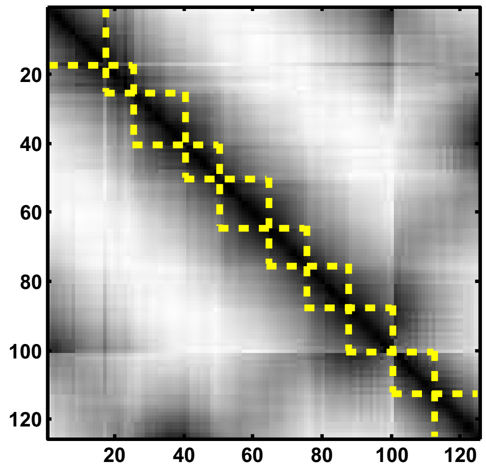
Figure A.14 demonstrates ReSL on the 3-dimensional data set first shown in Fig. A.11. ReSL is successful at partitioning the row objects into the visually preferred 5 clusters. However, similar to the Three Lines example in Example 4.1.3, CLODD is unable to partition the column objects. However, the question remains: How many clusters are there in the column objects? Depending on context, one could say that there is 1 cluster—all the objects in the 'figure 8'—or 5 clusters—each line that crosses the 5 row objects clusters. ReSL finds 10 clusters in the column objects, as shown in view (b), and 6 clusters in the union of the row and column objects, shown in view (c). Thus, the rule of thumb indicates $9 = 5 + 10 - 6$ co-clusters. The degree of co-clusterness matrix in view (e) has 10 "large" values (≥ 0.7). However, notice that these 10 "large" values correspond to 5 different connected-locations in $\mathbf{D}^*$. Thus, I believe that ReSL is successfully showing the co-clusters in this data.

What happens if I instead use ReSL on the co-iVAT images of the 3-dimensional data set from Fig. A.11. Figure A.15 demonstrates this result. View (a) shows that, again, ReSL is easily able to compute the partition of the row objects, which is expected given the quality of the co-iVAT image of $\mathbf{D}_r'^*$. View (b) shows that ReSL partitions the column objects into 9 clusters. Note that the 2 clusters, objects $\{1-25\}$ and $\{51-75\}$ are the top and bottom of the 'figure 8' pattern—see Fig. A.8(a). The union of the objects is partitioned into 6 clusters, as shown in view (c). Finally, the partition of the rectangular data $\mathbf{D}'^*$ is shown in view (d). The 2 large co-clusters—the first composed of row objects $\{101 - 150\}$ and column objects $\{51 - 75\}$, and the second composed of row objects $\{151 - 200\}$ and column objects $\{1 - 25\}$—are the 2 crosses formed by the objects at the top and bottom of the 'figure 8'. Finally, examining the degree of co-clusterness matrix in view (e) shows that there are 5 "large" values (≥ 0.6) in the matrix, as shown by the bold values.

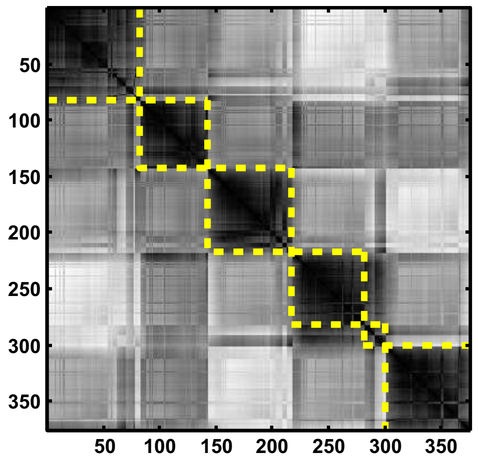
In summary, I believe the examples shown here offer evidence that ReSL and co-iVAT can be effective at showing the co-clusterness relations in rectangular data. However, it is my opinion that one should use ReSL on both the co-VAT and co-iVAT images and visually judge the evidence to choose the preferable result.



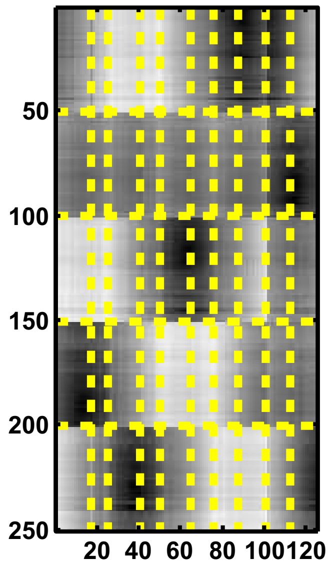
(a) Partition of $\mathbf{D}'^*_r$



(b) Partition of $\mathbf{D}'^*_c$



(c) Partition of $\mathbf{D}'^*_{r\cup c}$

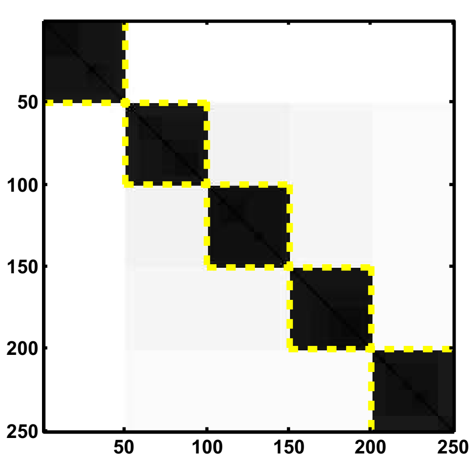


(d) Partition of $\mathbf{D}'^*$

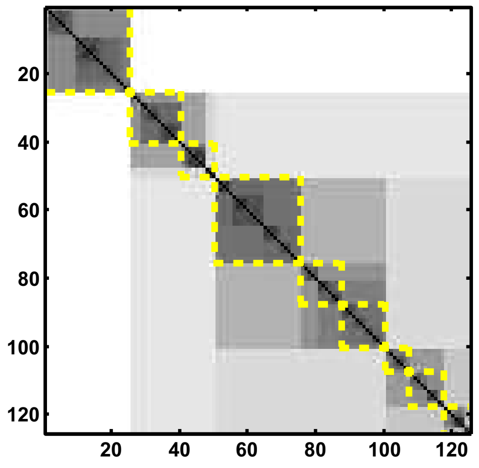
$$\begin{bmatrix} 0.4 & 0.1 & 0 & 0 & 0.2 & 0.4 & \mathbf{0.7} & \mathbf{0.7} & \mathbf{0.7} & 0.3 \\ 0.5 & 0.4 & 0.5 & 0.4 & 0.5 & 0.4 & 0.4 & 0.5 & 0.3 & \mathbf{0.7} \\ 0 & 0 & 0.2 & 0.4 & \mathbf{0.7} & \mathbf{0.7} & 0.4 & 0.2 & 0.4 & 0.4 \\ \mathbf{0.7} & \mathbf{0.7} & 0.4 & 0.1 & 0 & 0 & 0.1 & 0.4 & 0.4 & 0.5 \\ 0.2 & 0.4 & \mathbf{0.7} & \mathbf{0.7} & 0.4 & 0.1 & 0 & 0 & 0.3 & 0.6 \end{bmatrix}$$

(e) Partition values of $\mathbf{U}_{co}$

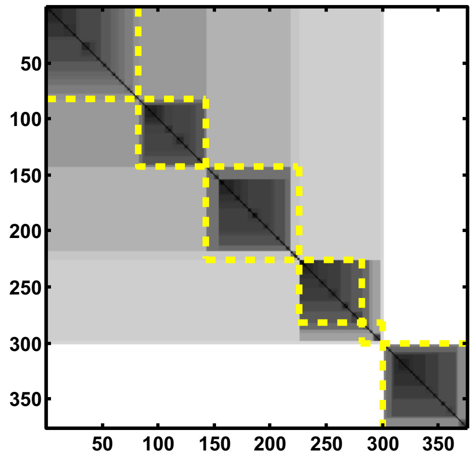
Figure A.14: ReSL partitions of co-iVAT reordered dissimilarity images shown in Fig. A.8



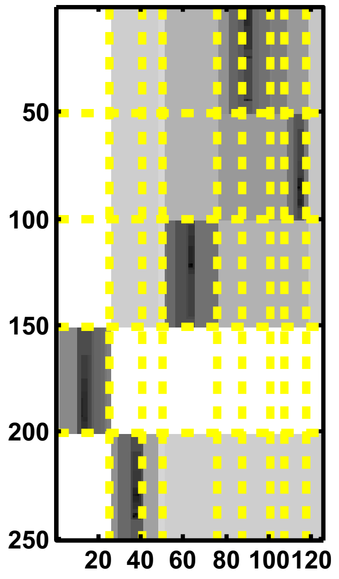
(a) Partition of $\mathbf{D}_r^*$



(b) Partition of $\mathbf{D}_c^*$



(c) Partition of $\mathbf{D}_{r\cup c}^*$



(d) Partition of $\mathbf{D}^*$

$$\begin{bmatrix} 0 & 0.2 & 0.2 & 0.3 & 0.5 & \mathbf{0.6} & 0.5 & 0.4 & 0.2 \\ 0 & 0.2 & 0.2 & 0.3 & 0.4 & 0.4 & 0.4 & \mathbf{0.6} & 0.2 \\ 0 & 0.2 & 0.2 & \mathbf{0.6} & 0.3 & 0.3 & 0.3 & 0.3 & 0.2 \\ \mathbf{0.6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{0.6} & 0.3 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix}$$

(e) Partition values of $\mathbf{U}_{co}$

Figure A.15: ReSL partitions of co-VAT reordered dissimilarity images shown in Fig. A.11

Appendix B

What is a Co-Cluster?

Many of the rectangular relational data examples given in this dissertation beckon a question of what objects comprise a co-cluster. So then, what is a co-cluster? There are many candidate answers to this questions. The reason I limit this discussion to the last pages of my dissertation is because one could easily explore this question and write a entire dissertation themselves. Thus, in this section, I introduce some of the existing discussion on this question and end with some conjecture and questions.

The term two-mode clustering was first coined by Mirkin [1996], although the process itself is first attributed to Hartigan [1975]. Biclustering, co-clustering, and two-mode clustering all refer to the process of clustering rectangular data, where the row objects and column objects are disjoint. Madeira and Oliveira [2004] identify four types of co-clusters (biclusters), illustrated in Fig. B.1:

1. Co-clusters with constant values, as in view (a);
2. Co-clusters with constant values on rows and columns, as in views (b,c);
3. Co-clusters with coherent values, as in view (d);
4. Co-clusters with coherent evolutions, as in view (e).

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

(a) Constant bicluster

1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

(b) Constant rows

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

(c) Constant columns

1	2	0	4
2	3	1	5
4	5	3	7
5	6	4	9

(d) Coherent values (additive model)

70	13	19	10
20	5	10	4
30	20	29	15
90	30	80	6

(e) Coherent evolution

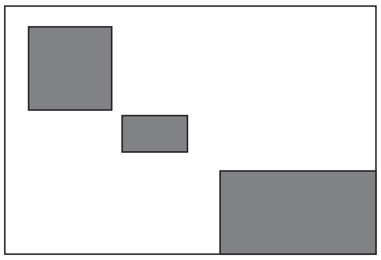
Figure B.1: Examples of different types of biclusters [Madeira and Oliveira, 2004].

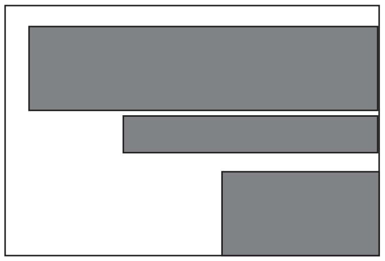
The first three types involve the values of the matrix directly. The coherent evolution types treats the matrix elements as symbols and examines the behavior of the symbols across the rows and columns. For example, in view (e), the behavior of this co-cluster is in the change of the values from column to column. From column 1 to column 2, the change in value is negative; from column 2 to column 3, the change in value is positive, and from column 3 to column 4 the change in value is negative. In this dissertation, I, for the most part, only deal with relational data; hence, I argue that the last three co-cluster types do not make sense. Only the first type, constant values, makes sense to me because this would indicate a group of equally similar (or dissimilar) objects. Furthermore, the only interesting co-clusters in relational data would be those that are grouped by a very small dissimilarity (highly similar objects) or very large dissimilarity (highly dissimilar objects).

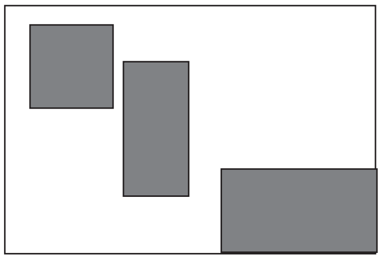
The co-clustering picture does not end there, however. VanMechelen et al. [2004] and Madeira and Oliveira [2004] describe different shapes of clusters. Partitions can take three general forms, which are enumerated in Table B.1. Moreover, in rectangular data these three types can be combined to produce hybrid instantiations. Figure B.2 gives examples of different cluster shapes.

Table B.1: Forms of cluster partitions [VanMechelen et al., 2004]

1. The clusters can consist of non-empty, non-intersecting subsets of objects;
2. Nested clusters are intersecting subsets of clusters (an example is hierarchical clusters);
3. Overlapping clusters are intersecting, non-nested clusters.


(a) Non-itersecting co-clusters


(b) Non-itersecting rows, nested columns co-clusters


(c) Non-itersecting columns, overlapping rows co-clusters

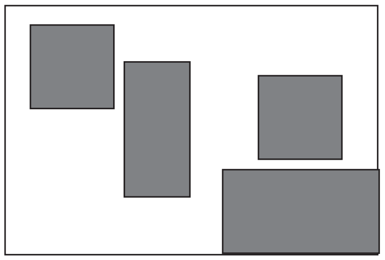
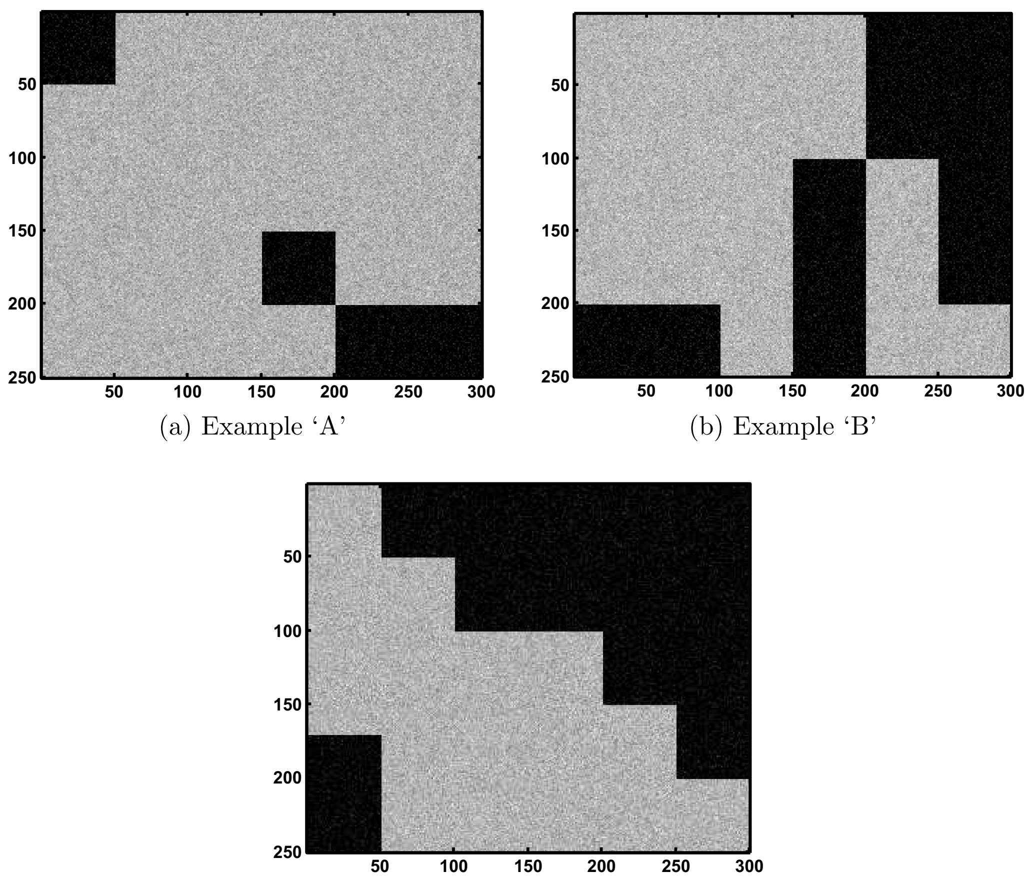

(d) Overlapping co-clusters

Figure B.2: Co-cluster shapes in dissimilarity data [VanMechelen et al., 2004]

Now that we have some formal definitions, let us examine some examples from this dissertation. Figure B.3 revisits the most compelling examples. Clearly, 'A' shows a cut-and-dry example of three non-intersecting co-clusters. However, 'B' and 'C' demand more inspection.

A hand-reordered version of Example 'B' is shown in Fig. B.4(a). As this view depicts, the dissimilarity data comprises 3 non-intersecting row clusters, labeled R1, R2, and R3, and 3 overlapping column clusters, C1, C2, and C3. One might argue

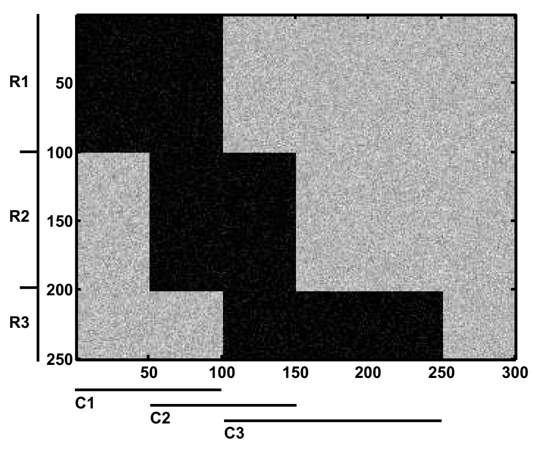

(a) Example 'A' (b) Example 'B'

(c) Example 'C'

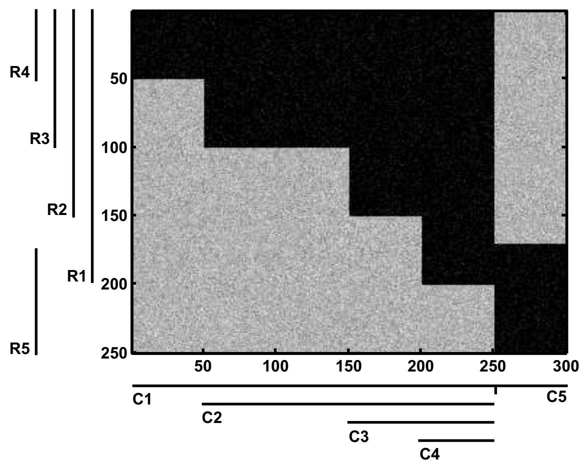
Figure B.3: Dissimilarity data sets that address the question: What is a co-cluster?

that column objects indexed 250-300 are another column cluster; however, I dissent on this opinion because these columns indicate objects that are not "close" to any of the row objects. Thus, not much is known about column objects 250-300. Certainly, their similarity to one another cannot be imputed from these data; the data are just noise.

Common opinion [VanMechelen et al., 2004, Madeira and Oliveira, 2004] calls the arrangement in Example 'B' three co-clusters. The ReSL degree of co-clusterness matrix, rearranged in the order of view (a) in B.4, is



(a) Example ‘B’ reordered



(b) Example ‘C’ reordered

Figure B.4: Cluster structures of rectangular dissimilarity data - (a) illustrates non-intersecting row clusters and overlapping column clusters, (b) illustrates nested and overlapping row clusters and nested and non-intersecting column clusters.

$$\begin{bmatrix} 0.96 & 0.96 & 0 & 0 & 0 \\ 0 & 0.96 & 0.96 & 0 & 0 \\ 0 & 0 & 0.96 & 0.96 & 0 \end{bmatrix}.$$

The three rows of this matrix correspond to the three non-intersecting row clusters. The five columns of this matrix clearly display the overlapping structure of the column structures, as well as the “empty” column cluster in the fifth column.

Now, let’s examine Example ‘C’ in Fig. B.4(b). This shows four nested row clusters with one overlapping row cluster and four nested column clusters with one non-intersecting column cluster. The number of co-clusters in this example is not clearly defined by any known reference. I believe that these data represent two co-clusters: one nested co-cluster, composed of row objects 1-200 and column objects 1-250 and an overlapping cluster composed of row objects 175-250 and column objects 251-300. The ReSL degree of co-clusterness matrix, arranged in the order of the view in B.4, is

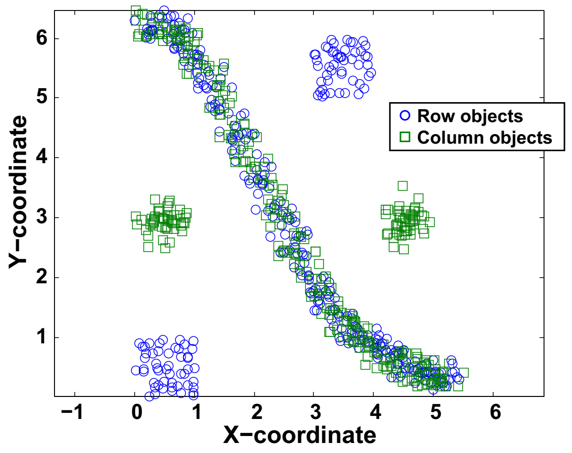
$$\begin{bmatrix} 0.96 & 0.96 & 0.96 & 0.96 & 0 \\ 0 & 0.96 & 0.96 & 0.96 & 0 \\ 0 & 0 & 0.96 & 0.96 & 0 \\ 0 & 0 & 0 & 0.96 & 0 \\ 0 & 0 & 0 & 0.96 & 0.96 \\ 0 & 0 & 0 & 0 & 0.96 \end{bmatrix}.$$

Again, ReSL clearly is able to show the nested structure of the clusters as well as the overlapping cluster.

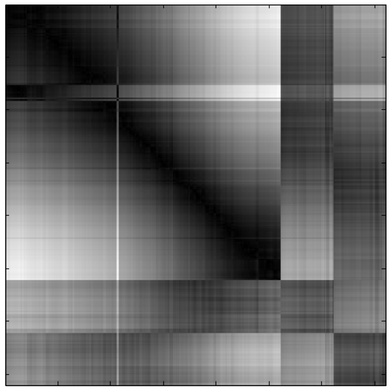
These examples illustrate the three types of clusters outlined by [VanMechelen et al., 2004]. However, they still do not get to the heart of the question, what is a co-cluster? They merely give some definitions. So I give one more example, shown in Fig. B.5, which may shed some light on the issue.

The plot in Fig. B.5(a) illustrates object data that comprise two circular clusters composed of just row objects and two circular clusters composed of just column objects and one curved-line co-cluster. I believe we all can agree that the curved line is the only co-cluster in this data set and comprises one co-cluster. However, the alternate co-VAT image in view (b) does not show this co-cluster as a dark block, as we saw in previous examples. But the co-iVAT image clearly shows a dark block, the normal representation of a co-cluster.

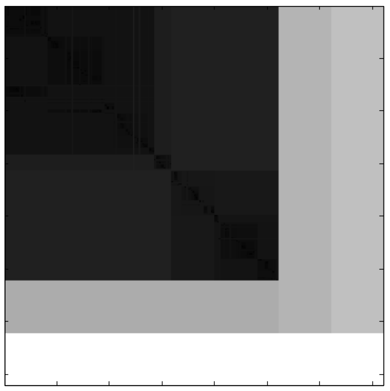
Thus, in summary, I believe that clustering in rectangular relational data brings up the same issues as clustering in object data or square relational data. Namely, that the definition of a cluster is context and problem dependent. End users cannot blindly apply clustering algorithms without regard; a knowledge of the underlying theory must be understood. Furthermore, clustering results are open to interpretation, which is why I believe that visual algorithms, such as those presented in this dissertation, are very useful. Humans (and, perhaps, animals) have an unmatched ability to recognize patterns and group these patterns in a meaningful way. The strength of



(a) Object Data



(b) Alternate-reordering co-VAT image - $\mathbf{D}^*$



(c) Alternate-reordering co-iVAT image - $\mathbf{D}'^*$

Figure B.5: coVAT and co-iVAT images of 360 row objects and 360 column objects represented by rectangular dissimilarity data

visual clustering algorithms is that the most information possible is presented to the end user, with suggestions as to how the data should be grouped.

Have I answered the question, what is a co-cluster? Of course not, because, ultimately, it all depends on what you are looking for.

What we see depends mainly on what we look for.
-Sir John Lubbock

References

MeSH. http://www.nlm.nih.gov/mesh/meshhome.html.

PubMed. http://www.ncbi.nlm.nih.gov/PubMed.

Merriam-webster online dictionary, 2009.

M.R. Anderberg. *Cluster Analysis for Applications*. Academic Press, Inc., New York, NY, 1973.

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

A. Asuncion and D.J. Newman. UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html, 2007.

G.H. Ball and D.J. Hall. ISODATA, a novel method of data analysis and classification. Technical report, Stanford University, Stanford, CA, 1965.

R. Baumgartner, R. Somorjai, R. Summers, W. Richter, and L. Ryner. Correlator beware: Correlation has limited selectivity for fMRI data analysis. *NeuroImage*, 12:240–243, 2000.

R. Baumgartner, R. Somorjai, R. Summers, and W. Richter. Ranking fMRI time courses by minimum spanning trees: Assessing coactivation in fMRI. *NeuroImage*, 13:734–742, 2001.

J.C. Bezdek. *Pattern Recognition With Fuzzy Objective Function Algorithms*. Plenum, New York, 1981.

J.C. Bezdek and R.J. Hathaway. VAT: A tool for visual assessment of (cluster) tendency. In *Proc. IJCNN*, pages 2225–30, Honolulu, HI, 2002.

J.C. Bezdek, J.M Keller, R. Krishnapuram, and N.R. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer, Norwell, 1999.

J.C. Bezdek, R.J. Hathaway, and J.M. Huband. Visual assessment of clustering tendency for rectangular dissimilarity matrices. *IEEE Trans. Fuzzy Systems*, 15 (5):890–903, October 2007.

W. Bo and R. Nevatia. Cluster boosted tree classifier for multi-view, multi-pose object detection. In *Proc. ICCV*, October 2007.

R.G. Cameron and G. Feuer, editors. *Apoptosis and Its Modulation by Drugs*. Handbook of Experimental Pharmacology. Springer, New York, NY, 2000.

L. Chen, X. Xu, and Y. Chen. An adaptive ant colony clustering algorithm. In *Proc. of the 3rd Int. Conf. Machine Learning and Cybernetics*, pages 1387–1392, Shanghai, China, August 2004.

H. Chernoff. The use of faces to represent points in k-dimensional space. *J. Amer. Stat. Assoc.*, 68:361–368, 1973.

M. Clerc and J. Kennedy. The particle swarm—explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. on Evolut. Comput.*, 6(1), February 2002.

W.S. Cleveland. *Visualizing Data*. Hobart Press, Summit, NJ, 1993.

T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 3 edition, 2009.

M. Cottrell, B. Hammer, A. Hasenfuss, and T. Villmann. Batch and median neural gas. *Neural Networks*, 19:855–863, 2006.

M. desJardins, J. MacGlashan, and J. Ferraioli. Interactive visual clustering. In *Proc. Int. Conf. Intelligent User Interfaces*, pages 361–364, Honolulu, HI, 2007.

I. Dhillon, D. Modha, and W. Spangler. Visualizing class structure of multidimensional data. In S. Weisberg, editor, *Proc. 30th Symp. on the Interface, Computing Science, and Statistics*, 1998.

I.S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery Data Mining*, pages 269–274, San Francisco, CA, 2001.

I.S. Dhillon, S. Mallela, and D.S. Modha. Information-theoretic co-clustering. In *Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery Data Mining*, pages 89–98, Washington, DC, 2003.

Y. Ding and R.F. Harrison. Relational visual cluster validity (RVCV). *Pattern Recognition Letters*, 28:2071–2079, 2007.

R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, second edition, October 2000.

J.C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *J. of Cybernetics*, 3(3):32–57, 1974.

A.P. Engelbrecht. *Computational Intelligence: An Introduction.* Wiley and Sons, Hoboken, NJ, 2 edition, 2007.

S.S. Epp. *Discrete Mathematics with Applications.* Brooks/Cole Publishing Company, Boston, MA, 2004.

B.S. Everitt. *Graphical Techniques for Multivariate Data.* Elsevier, North Holland, New York, 1978.

B. Fisher, T. Zoller, and J. Buhmann. Path based pairwise data clustering with application to texture segmentation. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2134:235–250, 2001.

G.D. Floodgate and P.R. Hayes. The Adansonian taxonomy of some yellow pigmented marine bacteria. *J. Gen. Microbiol.*, 30:237–244, 1963.

H. Frigui. *Advances in Fuzzy Clustering and Feature Discrimination with Applications*, chapter Simultaneous Clustering and Feature Discrimination with Applications, pages 285–312. John Wiley and Sons, 2007.

B. Fritzke. Let it grow—self-organizing feature maps with problem dependent cell structure. In *Proc. of ICANN*, pages 403–408, North Holland, Amsterdam, 1991.

T.O. Garnett and P.J. Duerksen-Hughes. Modulation of apoptosis by human papillomavirus (hpv) oncproteins. *Archives of Virology*, 151(12):2321–2335, December 2006.

A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, 1981.

M. Girolami. Mercer kernel-based clustering in feature space. *IEEE Trans. Neural Networks*, 13:780–784, 2002.

J.C. Gower and G.J.S. Ross. Minimum spanning trees and single linkage cluster analysis. *Appl. Statistics*, 18:54–64, 1969.

R. Gtz, S. Wiese, Shinichi Takayama, Guadalupe C. Camarero, Wilfried Rossoll, Ulrich Schweizer, Jakob Troppmair, Sibylle Jablonka, Bettina Holtmann, John C. Reed, Ulf R. Rapp, and Michael Sendtner. Essential role of bag-1 in differentiation and survival of hematopoietic and neuronal cells. *Nat Neurosci.*, 8(9):1169–1178, 2005.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. of Machine Learning and Research*, 3:1157–1182, March 2003.

B. Hammer, A. Micheli, A. Sperduti, and M. Strickert. Recursive self-organizing network models. *Neural Networks*, 17(8-9):1061 – 1085, 2004.

J. Handl, J. Knowles, and M. Dorigo. On the performance of ant-based clustering. *Frontiers in Artificial Intelligence and Applications*, 104:204–213, 2003.

J. Handl, J. Knowles, and M. Dorigo. Ant-based clustering and topographic mapping. *Artificial Life*, 12(1), 2005.

F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 2004.

J. Hartigan. *Clustering Algorithms*. Wiley, New York, 1975.

J.A. Hartigan and M.A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.

A. Hasenfuss and B. Hammer. Relational topographic maps. In *Advances in Intelligent Data Analysis VII*, pages 93–105. Springer Berlin / Heidelberg, 2007.

R.J. Hathaway and J.C. Bezdek. NERF c-MEANS: Non-euclidean relational fuzzy clustering. *Pattern Recognition*, 27:429–437, 1994a.

R.J. Hathaway and J.C. Bezdek. An iterative procedure for minimizing a generalized sum-of-squared-errors clustering criterion. *Neural, Parallel and Scientific Computations*, 2:1–16, 1994b.

R.J. Hathaway and J.C. Bezdek. Visual cluster validity for prototype generator clustering models. *Pattern Recognition Letters*, 24:1563–1569, 2003.

R.J. Hathaway, J.W. Davenport, and J.C. Bezdek. Relational duals of the c-means clustering algorithms. *Pattern Recognition*, 22(2):205–212, 1989.

R.J. Hathaway, J.C. Bezdek, and J.M. Huband. Scalable visual asseessment of cluster tendency for large data sets. *Pattern Recognition*, 39(7):1315–1324, July 2006.

T.C. Havens, J.M. Keller, E. MacNeal Rehrig, H.M. Appel, M. Popescu, J.C. Schultz, and J.C. Bezdek. Fuzzy cluster analysis of bioinformatics data composed of microarray expression data and Gene Ontology annotations. In *Proc. NAFIPS*, pages 1–6, New York, NY, 2008.

F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis*. Wiley, 1999.

J.R.M. Hosking. L-moments: analysis and estimation of distributions using linear combinations of order statistics. *J. of the Royal Statistical Society, Series B*, 52: 105–124, 1990.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *J. of Educational Psychology*, 24:417–441,498–520, 1933.

X. Hu. Integration of cluster ensemble and text summarization for gene expression analysis. In *Proc. BIBE*, pages 251–258, Taichung, Taiwan, May 2004.

J.M. Huband and J.C. Bezdek. *Computational Intelligence: Research Frontiers*, chapter VCV2 - Visual Cluster Validity, pages 293–308. Springer, Berlin / Heidelberg / New York, June 2008.

J.M. Huband, J.C. Bezdek, and R.J. Hathaway. Revised visual assessment of (cluster) tendency (reVAT). In *Proc. NAFIPS*, pages 101–104, Banff, Canada, June 2004. IEEE Press.

J.M. Huband, J.C. Bezdek, and R.J. Hathaway. bigVAT: visual assessment of cluster tendency for large data sets. *Pattern Recognition*, 38(11):1875–1886, November 2005.

T.J.P. Hubbard, B.L. Aken, S. Ayling, B. Ballester, K. Beal, E. Bragin, S. Brent, Y. Chen, P. Clapham, L. Clarke, G. Coates, S. Fairley, S. Fitzgerald, Fernandez J. Banet, L. Gordon, S. Graf, S. Haider, M. Hammond, R. Holland, K. Howe, A. Jenkinson, N. Johnson, A. Kahari, D. Keefe, S. Keenan, R. Kinsella, F. Kokocinski, E. Kulesha, D. Lawson, I. Longden, K. Megy, P. Meidl, B. Overduin, A. Parker, B. Pritchard, D. Rios, M. Schuster, G. Slater, D. Smedley, W. Spooner, G. Spudich, S. Trevanion, A. Vilella, J. Vogel, S. White, S. Wilder, A. Zadissa, E. Birney, F. Cunningham, V. Curwen, R. Durbin, Fernandez X. M. Suarez, J. Herrero, A. Kasprzyk, G. Proctor, J. Smith, S. Searle, and P. Flicek. Ensembl 2009. *Nucl. Acids Res.*, 37:D690–D697, 2009.

L.J. Hubert. Some applications of graph theory to clustering. *Psychometrika*, 39(3): 283–309, September 1974.

A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

J.J. Jiang and D.W. Conrath. Semantic similarity based on corpus statistics and lexical ontology. In *Proc. of Int. Conf. Res. on Comp. Linguistics X*, Taiwan, 1997.

D.A. Johnson and D.W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, Englewood Cliffs, NJ, 6 edition, 2007.

S.C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 2:241–254, 1967.

P. Kankar and S. Mukherjea. Text-based summarization and visualization of gene clusters. In *Proc. ACM Symposium on Applied Computing*, pages 210–211, Santa Fe, NM, March 2005.

G. Karypis, E.-H. Han, and V. Kumar. CHAMELEON: a hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32.

S. Kaski and T. Kohonen. Exploratory data analysis by the self-organizing map: Structures of welfare and poverty in the world. In *Proc., 3rd Int. Conf. on Neural Networks in the Capital Markets*, pages 498–507, London, England, 1996.

S. Kaski, J. Kangas, and T. Kohonen. Bibliography of self-organizing map (SOM) papers: 1981-1997. *Neural Computing Surveys*, 1:102–350, 1998.

J.M. Keller, M. Popescu, and J.A. Mitchell. Taxonomy-based soft similarity measures in bioinformatics. In *Proc. IEEE Int. Conf. on Fuzzy Systems*, pages 23–30, Budapest, Hungary, July 2004. IEEE.

J.M. Keller, J.C. Bezdek, M. Popescu, N. Pal, J. Mitchell, and J. Huband. Gene ontology similarity measures based on linear order statistics. *Int. J. on Uncertainty, Fuzziness and Knowledge-Based Systems*, 14(6):639–661, 2006.

S. Khan, G. Situ, K. Decker, and C.J. Schmidt. GoFigure: Automated Gene Ontology annotation. *Bioinf.*, 19(18):2484–2485, 2003.

B. Kleiner and J.A. Hartigan. Representing points in many dimensions by trees and castles. *J. Amer. Stat. Asooc.*, 76:260–269, 1981.

G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications.* Prentice Hall, Upper Saddle River, New Jersey, 1995.

T. Kohonen. Automatic formation of topological maps of patterns in a self-organizing system. In E. Oja and O. Simula, editors, *Proc. of SCIA*, pages 214–220, Helsinki, Finland, 1981.

T. Kohonen. *Self-Organizing Maps*, volume 30 of *Information Sciences.* Springer, Berlin, 2001.

T. Konkela, S. Kaski, K. Lagus, and T. Kohonen. Exploration of full-text databases with self-organizing maps. In *Proc. ICNN*, volume 1, pages 56–61, Washington, D.C., June 1996.

H.P. Kriegel, P. Kroger, and A. Zimek. Clustering high dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowledge Discovery from Data*, 3(1):1–58, March 2009.

R. Krishnapuram and J.M. Keller. A possibilistic approach to clustering. *IEEE Trans. on Fuzzy Sys.*, 1(2), May 1993.

R. Krisnapuram, H. Frigui, and O. Nasroui. Fuzzy and possibilistic shell clustering algorithms and their applications to boundary detection and surface approximations. *IEEE Trans. Fuzzy Systems*, 3(1):29–60, February 1995.

J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proc. of the Am. Math. Soc.*, volume 7, pages 48–50, February 1956.

J.B. Kruskal and M. Wish. *Multidimensional Scaling.* Sage Publications, Beverly Hills, CA, 1978.

K. Kummamuru, A. Dhawale, and R. Krishnapuram. Fuzzy co-clustering of documents and keywords. In *Proc. IEEE Int. Conf. Fuzzy Systems*, pages 772–777, St. Louis, MO, 2003.

R. Kurzweil. *The Singularity is Near.* Viking Penguin, USA, September 2006.

C. Leacock and M. Chodorow. *WordNet: An electronic lexical database*, chapter Combining local context and WordNet similarity for word sense identification, pages 265–283. MIT Press, 1998.

D. Lin. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Maching Learning*, pages 296–304, San Francisco, CA, 1998. Morgan Kaufmann.

R.F. Ling. A computer generated aid for cluster analysis. *Communications of the ACM*, 16(6):355–361, 1973.

H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*, volume 454 of *Springer Int. Series in Eng. and Comput. Sci.* Springer, 1998.

H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowledge and Data Engineering*, 17(4):491–502, April 2005.

B. Long, Z. Zhang, and P.S. Yu. A probabilistic framework for relational clustering. In *Proc. KDD*, San Jose, CA, August 2007.

P.W. Lord, R.D. Stevens, A. Brass, and C.A. Goble. Semantic similarity measure as a tool for exploring the gene ontology. *Pacific Symposium on Biocomputing*, pages 601–612, 2003.

S. Madeira and A. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE Trans. Computational Biology and Bioinformatics*, 1(1):24–45, January 2004.

T. M. Martinez, S. G. Berkovich, and K. J. Shulten. Neural-gas network for vector quantization and its applications to time-series prediction. *IEEE Trans. on Neural Networks*, 4(4):558–569, 1993.

J. McCarthy, M.L. Minsky, N. Rochester, and C.E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, August 1955.

Dylan Loeb McClain. Once again, machine beats human champion at chess. *The New York Times*, December 2006.

B. Mirkin. *Mathematical Classification and Clustering.* Kluwer Academic Press, Dordrecht, The Netherlands, 1996.

J. Myllyharju and K.I. Kivirikko. Collagens, modifying enzymes, and their mutation in humans, flies, and worms. *Trends in Genetics*, 20(1):33–43, 2004.

A. Nagar and H. Al-Mubaid. A new path length measure based on go for gene similarity with evaluation using SGD pathways. In *Proc. IEEE Int. Symp. Computer-Based Medical Systems*, pages 590–595, June 2008.

Svetlana I. Novikova, Fang He, Jie Bai, Irina Badan, Irina A. Lidow, and Michael S. Lidow. Cocaine-induced changes in the expression of apoptosis-related genes in the fetal mouse cerebral wall. *Neurotoxicology and Teratology*, 27:3–14, 2005.

M. Oja, S. Kaski, and T. Kohonen. Bibliography of self-organizing map (SOM) papers: 1998-2001 addendum. *Neural Computing Surveys*, 3:1–156, 2003.

N.R. Pal, K. Pal, and J.C. Bezdek. A mixed c-means clustering model. In *Proc. IEEE Int. Conf. Fuzzy Systems*, pages 11–21, Spain, 1997.

N.R. Pal, K. Pal, J.M. Keller, and J.C. Bezdek. A possibilistic fuzzy c-means clustering algorithm. *IEEE Trans. Fuzzy Systems*, 13(4):517–530, August 2005.

L.A.F. Park, J.C. Bezdek, and C.A. Leckie. Visualization of clusters in very large rectangular dissimilarity data. In G. Sen Gupta and S.C. Mukhopadhyay, editors, *Proc. 4th Int. Conf. Autonomous Robots and Agents*, pages 251–256, February 2009.

W. Pedrycz. *Computational Intelligence: An Introduction.* CRC Press, Inc., Boca Raton, FL, 1 edition, 1997.

M. Pöllä, T. Honkela, and T. Kohonen. Bibliography of self-organizing map (SOM) papers: 2002-2005 addendum. forthcoming, *Neural Computing Surveys*, 2006.

D. Poole, A. Mackworth, and R. Goebel. *Computational Intelligence: A Logical Approach.* Oxford University Press, New York, NY, 1998.

M. Popescu, J.M. Keller, J.A. Mitchell, and J.C. Bezdek. Functional summarization of gene product clusters using Gene Ontology similarity measures. In *Proc. 2004 ISSNIP*, pages 553–559, Piscataway, NJ, 2004. IEEE Press.

M. Popescu, J.M. Keller, and J.A. Mitchell. Fuzzy measures on the Gene Ontology for gene product similarity. *IEEE Trans. on Computational Biology and Bioinformatics*, 3(3):263–274, 2006.

M. Popescu, J.C. Bezdek, J.M. Keller, T.C. Havens, and J.M. Huband. A new cluster validity measure for bioinformatics relational datasets. In *Proc. FUZZ-IEEE*, pages 726–731, Hong Kong, China, June 2008.

R.C. Prim. Shortest connection networks and some generalisations. *Bell System Tech. J.*, 36:1389–1401, 1957.

S. Raychaduri and R.B. Altman. A literature-based method for assessing the functional coherence of a gene group. *Bioinformatics*, 19(3), 2003.

G. Reinelt. *Lecture Notes in Computer Science*, chapter The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag, Berlin, 1994.

P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *International Joint Conference for Artificial Intelligence*, pages 448–453, 1995.

H. Ritter and T. Kohonen. Self-organizing semantic maps. *Biological Cybernetics*, 61 (4):241–254, August 1989.

D.J. Rosenkrantz, R.E. Stearns, and P.M. Lewis. An analysis of several heuristics for the traveling salesman problem, 1977.

Rahila H. Sheikh, M.M. Raghuwanshi, and Anil N. Jaiswal. Genetic algorithm based clustering: A survey. *Emerging Trends in Engineering and Technology, International Conference on*, 0:314–319, 2008. doi: http://doi.ieeecomputersociety.org/10.1109/ICETET.2008.48.

J. Skillings. Getting machines to think like us. *CNET News*, July 2006.

I. Sledge, J.M. Huband, and J.C. Bezdek. (Automatic) Cluster count extraction from unlabeled datasets. *Proc. FSKD*, 1:3–13, 2008.

I. Sledge, T.C. Havens, J.M. Huband, J.C. Bezdek, and J.M. Keller. Finding the number of clusters in ordered dissimilarities. *Soft Computing*, 13(12):1125–1142, October 2009a.

I.J. Sledge, J.C. Bezdek, T.C. Havens, and J.M. Keller. Relational duals of cluster validity functions for the c-means family. *in review, Pattern Recognition*, 2009b.

I.J. Sledge, T.C. Havens, J.C. Bezdek, and J.M. Keller. Relational generalizations of validity indexes. *in review, IEEE Trans. Fuzzy Systems*, 2009c.

P.H.A. Sneath. A computer approach to numerical taxonomy. *J. Gen. Microbiol.*, 17: 201–226, 1957.

A. Strehl and J. Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *Proc. HiPC*, volume 1970, pages 525–536, Springer, NY, 2000a. LNCS.

A. Strehl and J. Ghosh. Value-based customer grouping from large retail data-sets. In *Proc. SPIE Conf. on Data Mining and Knowledge Discovery*, volume 4057, pages 33–42, Bellingham, WA, 2000b. SPIE Press.

M. Sugeno. *Theory of fuzzy integral and its applications.* PhD thesis, Tokyo Institute of Technology, 1974.

M. Sugeno. *Fuzzy Automata and Decision Processes*, chapter Fuzzy measures and fuzzy integrals: a survey, pages 89–102. North-Holland, New York, 1977.

H. Tahani and J.M. Keller. Information fusion in computer vision using the fuzzy integral. *IEEE Trans. Systems Man Cybernet.*, 20(3):741–773, 1990.

E. Tejada and R. Minghim. Improved visual clustering of large multi-dimensional data sets. In *Int. Conf. Information Visualisation*, pages 818–825, London, England, July 2005.

The Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Res.*, 32:D258–D261, 2004.

The UniProt Consotium. The universal protein resource (UniProt). *Nucleic Acids Res.*, 35:D193–D197, 2007.

S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Academic Press, San Diego, CA, 4 edition, 2009.

H. Timm and R. Kruse. A modification to improve possibilistic fuzzy cluster analysis. In *Proc. IEEE Int. Conf. Fuzzy Systems*, Honolulu, HI, 2002.

H. Timm, C. Borgelt, C. Doring, and R. Kruse. An extension to possibilistic fuzzy cluster analysis. *Fuzzy Sets and Systems*, 147:3–16, 2004.

T.D. Tran-Luu. *Mathematical concepts and novel heuristic methods for data clustering and visualization.* PhD thesis, U. of Maryland, College Park, MD, 1996.

R.C. Tryon. *Cluster Analysis.* Edwards Bros., Ann Arbor, MI, 1939.

J.W. Tukey. *Exploratory Data Analysis.* Addison-Wesley, Reading, MA, 1977.

A. Turing. Computing machinery and intelligence. *Mind*, 236:433–460, October 1950.

E. van Someren, L. Wessels, and M. Reinders. GENLAB toolbox. http://genlab.tudelft.nl, 2000.

I. VanMechelen, H. Bock, and P. DeBoeck. Two-mode clustering methods: a structured overview. *Statistical Methods in Medical Research*, 13:363–394, 2004.

L. Wang, C. Leckie, K. Rao, and J.C. Bezdek. Automatically determining the number of clusters from unlabeled data sets. *IEEE Trans. Knowledge Engr.*, 21(3):335–350, March 2009.

L. Wang, T.V.U. Nguyen, J.C. Bezdek, C.A. Leckie, and K. Ramamohanarao. iVAT and aVAT: enhanced visual analysis for cluster tendency assessment. In *Proc. PAKDD*, Hyderabad, India, June 2010.

Z. Wu and M. Palmer. Verb semantics and lexical selection. In *Proc. 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, 1994.

D. Xu, J.M. Keller, M. Popescu, and R. Bondugula. *Applications of Fuzzy Logic in Bioinformatics*. Imperial College Press, London, UK, 2008.

R. Xu and D.C. Wunsch II. *Clustering*. IEEE Press, Psicataway, NJ, 2009.

R.R. Yager and J. Kacprzyk. *The Ordered Weighted Averaging Operators: Theory and Applications*. Springer, Netherlands, May 1997.

D.Q. Zhang and S.C. Chen. Clustering incomplete data using kernel-based fuzzy c-means algorithm. *Neural Processing Letters*, 18:155–162, 2003.

Z. Zhang, X. Wu, and P.S. Yu. Spectral clustering for multi-type relational data. In *Proc. 23rd Int. Conf. on Machine Learning*, Pittsburgh, PA, 2006.

VITA

Timothy C. Havens was born in Traverse City, Michigan. He received the Doctorate of Philosophy in electrical and computer engineering from the University of Missouri in July, 2010. He graduated in August, 1999, with a Bachelor of Science degree in electrical engineering from Michigan Tech University (MTU) in Houghton, Michigan. In August, 2000, he was awarded a Master of Science degree in electrical engineering, also from MTU. Immediately following his graduate work at MTU, Tim joined the technical staff at MIT Lincoln Laboratory. At MIT, Tim focused on the simulation and modeling of *directed energy systems* and *global positioning systems* (GPS). He specializes in the theory and application of computational intelligence — including fuzzy sets and fuzzy systems, evolutionary methods, and swarm intelligence — pattern recognition, and signal and image processing in problems related to sensor fusion, informatics, ontologies, and computer-vision.

Tim will be joining the Biometrics Research Group at Michigan State University in September, 2010, as an NSF Computing Innovation Postdoctoral Fellow.

Tim is also an accomplished musician and has won outstanding musicianship and ensemble awards at national collegiate jazz festivals. He has recorded and performed music in multiple genres including jazz, afro-Cuban, rock, and funk. He has performed and recorded with many notable musicians, including Mike Mainieri, Jeff “Skunk” Baxter, Mike Irish, and Steve Zenz.