Copyright
by
Chase Serhur Krumpelman
2010

The Dissertation Committee for Chase Serhur Krumpelman
certifies that this is the approved version of the following dissertation:

Overlapping Clustering

Committee:

Joydeep Ghosh, Supervisor

Adnan Aziz

Orly Alter

Raymond Mooney

Sriram Viswanath

Overlapping Clustering

by

Chase Serhur Krumpelman, B.S.E.E.; M.S.E.

DISSERTATION
Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN
August 2010

Overlapping Clustering

Publication No. ____________________

Chase Serhur Krumpelman, Ph.D.
The University of Texas at Austin, 2010

Supervisor: Joydeep Ghosh

Analysis of large collections of data has become inescapable in many areas of scientific and commercial endeavor. As the size and dimensionality of these collections exceed the pattern recognition capability of the human mind computational analysis tools become a necessity for interpretation. Clustering algorithms, which aim to find interesting groupings within collections of data, are one such tool. Each algorithm incorporates into its design an inherent definition of "interesting" intended to capture nonrandom data groupings likely to have some interpretation to human users. Most existing algorithms include as part of their definition of "interesting" an assumption that each data point can belong at most to one grouping. While this assumption allows for algorithmic convenience and ease of analysis, it is often an artificial imposition on true underlying data structure. The idea of allowing points to belong to multiple groupings - known as "overlapping" or "multiple membership" clustering - has emerged in several domains in *ad hoc* solutions lacking conceptual unity in approach, interpretation, and analysis. This dissertation proposes general, domain-independent elucidations and practical techniques which address each of these.

We begin by positing overlapping clustering's role specifically, and clustering's role in general, as assistive technologic tools allowing human minds to represent and interpret structures in data beyond the capability of our innate senses. With this guiding purpose clarified, we provide a catalog of existing techniques. We then address the issue of objectively comparing the results of different algorithms, specifically examining the previously defined Omega index, as well as multiple membership generalizations of normalized mutual information. Following that comparison, we propose a novel approach to comparing clusterings called *cluster alignment*. By combining a sorting algorithm with a greedy matching algorithm, we produce comparably organized membership matrices and a means for both numerically and visually comparing multiple-membership assignments. With overlapping clustering's purpose defined, and the means to analyze results, we move on to presenting algorithms for efficiently discovering overlapping clusters in data. First, we present a generalization of one of the common themes in the *ad hoc* approaches: additive clustering. Starting with a previously developed structural model of additive clustering, we generalize it to be applicable to any regular exponential family distribution thereby extending its utility into several domains, notably high-dimensional sparse domains including text and recommender systems. Finally, we address overlapping clustering by examining the properties of data in *similarity spaces*. We develop a probabilistic generative model of overlapping data in similarity spaces, and then develop two conceptual approaches to discovering overlapping clustering in similarity spaces. The first of these is the conceptual multiple-membership generalization of hierarchical agglomerative clustering, and the second is an iterative density hill-climbing algorithm.

Table of Contents

Abstract **iv**

List of Tables **x**

List of Figures **xi**

Chapter 1. Introduction **1**

1.1 Problem Statement of Clustering 3

1.2 Notation 5

1.3 Objectives, Contributions, and Organization 6

Chapter 2. Background **8**

2.1 Single Membership Clustering 8

2.2 Multiple Membership Clustering 11

2.2.1 Soft Models 11

2.2.1.1 SVD-like matrix decompositions 12

2.2.1.2 Gene Shaving 14

2.2.2 Multiple-Membership Extensions to Hierarchical Agglomerative Clustering 15

2.2.2.1 Jardine-Sibson B-clustering and Articulation Point Cuts 15

2.2.2.2 Pyramid Hierarchical Clustering 16

2.2.3 Similarity-Space Additive Clusterings 18

2.2.3.1 Decomposing Similarity Graphs: MODES . . . 22

2.2.4 Feature Space Additive Clustering 23

2.2.4.1 The “Plaid” Model 24

2.2.4.2 The SBK Model: Decomposing Gene Expression into Cellular Processes 25

2.2.4.3 Biclustering 27

2.2.4.4 Bond-Energy and Rank-Order Clustering 28

Chapter 3. Comparison of Overlapping Clusterings 30
3.1 Single-Membership Evaluation 31
3.1.1 Desiderata 31
3.1.2 Probability Model of Single Membership Clustering 32
3.1.3 Mutual Information 33
3.1.4 Pairwise Mutual Information 35
3.1.5 The Rand Index and Adjusted Rand Index 36
3.1.6 ARI and PNMI 37
3.2 Multiple Membership Evaluation 38
3.2.1 A Probabilistic View of Multiple-membership Clustering 40
3.2.2 Mutual Information 42
3.2.3 Pairwise Mutual Information 43
3.2.4 The Multimembership Rand Index: The Omega Index 44
3.2.5 Omega Index and Multiple-membership PNMI 47

Chapter 4. Cluster Alignment 48
4.1 Introduction 48
4.2 Aligning Nondisjoint Membership Matrices with Hypergeometric p-values 48
4.3 Visualization of p-value based alignment 51
4.4 Comparison of Clustering Methods 52
4.5 Alignment for Cluster Ensembles 54
4.6 Conclusion 57

Chapter 5. Model Based Overlapping Clustering 61
5.1 Introduction 61
5.2 Models 63
5.2.1 The SBK Model 64
5.2.2 Overlapping Clustering by Thresholding a Mixture Model 66
5.2.3 The MOC Model 68
5.3 Algorithms and Analysis 70
5.3.1 Updating α 71
5.3.2 Updating M 71

5.3.3 Updating A 76
5.4 Experiments 77
5.4.1 Methodology 77
5.4.2 Results 79

Chapter 6. Similarity Space Overlapping Clustering 82
6.1 Introduction 82
6.2 Generative Model of Similarity Space Overlapping Clusters . . 83
6.2.1 Model Considerations and Background 83
6.2.2 The Similarity Space Multiple-Membership Generative Model 87
6.3 Algorithms for Finding Clusters in Similarity Spaces 89
6.4 Generalizing Complete-Link Hierarchical Clustering to Multiple Memberships 90
6.4.1 Definition of s-level, *elevated subset*, and *rise* 93
6.4.2 Generalizing complete-link hierarchical clustering 94
6.5 Existing Algorithms for Finding Elevated Subsets 95
6.5.1 Exact Algorithm for Enumerating Elevated Subsets . . . 97
6.5.2 Feasible Algorithm for Enumerating Elevated Subsets . 100
6.6 Similarity Space Density-Searching Multiple Membership Algorithms 102
6.6.1 Cluster Core Finding 105
6.6.2 Cluster Growth 105
6.6.2.1 Connectivity and Single-Link *vs* Complete-Link Character 106
6.6.3 Cluster Merging 107
6.6.4 Parameters 107
6.6.5 Multiple-Membership Density Hill Climbing Algorithm . 107
6.7 Experiments 108
6.7.1 Strawman for Comparison 108
6.7.2 Datasets 109
6.7.3 Results 110
6.7.4 Similarity Space Density Climbing Applied to Microarray Data 111
6.7.5 Discussion 113

Chapter 7. Conclusions and Future Directions **114**
7.1 Contributions 114
7.2 Future Work 115

Appendices **117**

Appendix A. Mutual Information when labels exceed datapoints **118**

Appendix B. Synthetic Datasets **119**
B.1 Synthetic Dataset 1 119
B.2 Synthetic Dataset 2 121
B.3 Synthetic Dataset 3 121

Bibliography **124**

Vita **132**

List of Tables

3.1 Terms used in PNMI and Rand Index calculations 36

3.2 Table for calculating multiple-membership pairwise mutual information and Omega Index 45

5.1 Comparison of results of MOC and thresholded mixture models on all datasets 80

5.2 Results: `KMS` vs Bounded Least Squares (with search) for synthetic data 80

6.1 Parameters of the similarity space clustering generative model. 88

6.2 Parameters of the example synthetic data set. 89

6.3 5 points similarity space used to create the multiple membership dendrogram shown in Figure 6.8. 95

6.4 Parameters for similarity space density hill-climbing algorithm. 107

6.5 Parameters used for experiments. 110

6.6 Results of algorithms compared to strawman on synthetic datasets. Because of its random initialization, results for the Density Climbing algorithm are averaged over ten runs. 110

B.1 Parameters of Synthetic Dataset 1 119

B.2 Parameters of Synthetic Dataset 2 121

B.3 Parameters of Synthetic Dataset 3 122

List of Figures

1.1 Venn diagram representing label assignments of points, and traditional clustering labelling of the same situation. Note that the Venn diagram allows points to be labelled as belonging to one, both, or no clusters, while the traditional clustering requires each point to belong to exactly one cluster. 3

2.2 Jardine-Sibson cluster for $b = 2$ 16

2.3 Pyramid hierarchical clustering of yeast genes from Aude, *et al.* [ADLCJ99] 18

3.1 The empirical relationship between ARI, PNMI, and NMI for a dataset of 200 points and 10 clusters as the shared information between labellings changes. The vertical axis indicates the value of the comparison measure. The horizontal axis indicates the distortion between the two label matrices, which each tick indicating 0.5% of the labels changed. 39

3.2 The empirical relationship between the Omega Index, PNMI, NMI, and aligned NMI for a dataset of 200 points labeled with 10 labels as the shared information between labellings changes. The vertical axis indicates the value of the comparison measure, and each step on the horizontal axis represents randomly assigning an additional 1% of the labels of the second clustering. 46

4.1 Visualization of correspondence between ground truth and MOC overlapping clustering for synthetic microarray data. The top frame is the "signature" of the ground truth, with the rows sorted as described in Section 4.3. The second frame is the "signature" of the MOC clustering, with the rows in the same order as in the top frame, and the columns matched using the algorithm described in Section 4.2. The final frame is a bar chart of the alignment p-values. The more negative the alignment p-values, the less likely the alignment happened due to random chance. 55

4.2 Visualization of correspondence between ground truth and Soft k-means overlapping clustering for synthetic microarray data. 56

4.3 Visualization of aligned cluster signatures of MOC, soft k-means, and gene shaving clustering results on the synthetic data. . . 58

4.4 Visualization of ground truth cluster labels and aligned majority-vote consensus of MOC, soft k-means and gene shaving. The consensus recovers the actual labelling better than any of the individual clusterers. 59

6.1 Histogram of the values in the similarity space of the iris dataset. 84

6.2 The upper row shows histograms of the within-cluster similarity and between-cluster similarity of points in the Iris dataset. Note that distribution of similarities of points not sharing a cluster has most of its mass near zero, while the distribution of similarities of points sharing a cluster has a center of mass far from zero. These two distributions can be approximated by beta distributions, shown in the second row. 84

6.3 Heatmap of the similarity space of Iris. Notably, the similarity space is a symmetric matrix with ones along the diagonal and a notable block structure. Each block along the diagonal is a region of high similarity between points, i.e., a cluster. The middle and lower right cluster correspond to *Iris versicolor* and *Iris virginica* which are not linearly separable groups. The off-axis similarity indicates high similarity between points in the two groups. 86

6.4 Values in the similarity space of a dataset containing clusters can be modeled as values drawn from one of two beta distributions. The distribution shown in blue with most of its probability mass close to zero represents the distribution of similarities of points not in the same cluster. We term this distribution the "background" and parameterize it as $\Theta_{background}$. The distribution shown in green with most of its probability mass close to one represents the distribution of similarities of highly similar points likely to be in the same cluster. We term this distribution the "cluster" distribution and parameterize it as $\Theta_{cluster}$. . . . 87

6.5 The "background" and "cluster" Beta functions with the parameters given in Table 6.2 used to generate the synthetic dataset. 90

6.6 Characteristics of the synthetic data set generated using the parameters in Table 6.2. The first pane shows the distribution of within-cluster similarities, the second shows the background similarity distribution, and the bottom pane the distribution of memberships. Note that the within-cluster distribution is a mixture of the specified within-cluster Beta distribution and a small amount of the background Beta distribution. This is due to the parameter c which specifies complete-link character being less than one: points within a cluster are not required to have high similarity to all other points within the cluster. . . . 91

6.7 Heatmap of the synthetic data set generated using the parameters in Table 6.2. Due to the overlapping nature of the similarity space, the matrix can not be reordered so that the high similarities form a block-diagonal matrix. 92

6.8 Dendrogram showing all elevated subsets and inheritance paths for the 5 point similarity space given in Table 6.3. Vertices corresponding to elevated subsets found by complete-link hierarchical clustering are circled in red. It is notable that the complete set of elevated subsets is a superset of the set of complete-link hierarchical clusters, and that the all-points cluster at the top and the singleton clusters at the bottom are shared between the algorithms. 96

6.9 For the point at the center of the figure, the sphere representing the core threshold t_{core} is shown in red and the sphere representing the peripheral threshold $t_{periphery}$ is shown in blue. The points composing the *C-neighborhood* are shown in red, those of the *P-neighborhood* in blue, and those of the ϕ*-neighborhood* in black. 104

6.10 Criteria for granting *P-neighbors* cluster membership. The point indicated in green is in the *P-neighborhood* of the point at the center of the cluster, and contains more than half of the cluster's members in its own *C-neighborhood*, therefore, it would be added to the cluster. The point indicated in cyan is also in the *P-neighborhood* of the cluster, but contains very few of the cluster's members in its *C-neighborhood* and therefore would not be added to the cluster. 106

6.11 Distributions of within-cluster similarity (top pane), between-cluster similarity (middle pane), and shared labels (bottom pane) of the clustering found by the similarity space density hill climbing algorithm. It is notable by comparing the top and middle panes that the discovered clusters have a much higher internal similarity than the background. The bottom pane shows that genes in this clustering belong to as many as 5 different clusters. 112

B.1 Characteristics of the synthetic data set generated using the parameters in Table B.1. The first pane shows the distribution of within-cluster similarities, the second shows the background similarity distribution, and the bottom pane the distribution of memberships. Note that the within-cluster distribution is a mixture of the specified within-cluster Beta distribution and a small amount of the background Beta distribution. This is due to the parameter c which specifies complete-link character being less than one: points within a cluster are not required to have high similarity to all other points within the cluster. . . . 120

B.2 Structure of Synthetic Dataset 2. The top pane shows the within-cluster distribution of similarities, the middle pane shows the distribution of background similarities, and the bottom pane the distribution over number of cluster memberships per point. 122

B.3 Structure of Synthetic Dataset 3. The top pane shows the within-cluster distribution of similarities, the middle pane shows the distribution of background similarities, and the bottom pane the distribution over number of cluster memberships per point. 123

Chapter 1

Introduction

It could be reasonably claimed that the primary goal of "Data Mining" is to replicate on data that is massive and high-dimensional what a human could do on data that is low in number and dimension. Modern technologies allow the collection of data that is routinely well beyond human analysis capabilities, leading us to a situation that has been described as "drowning in information but starved for knowledge"[Nai88]. To find this "knowledge" in the information deluge, humans have constructed tools: mathematical models and algorithms that attempt to replicate on beyond-human-scale data what a human would do on human-scale data.

Clustering - finding similar subsets of objects within a collection - is a fundamental analysis technique which humans perform naturally, but which becomes very difficult as the scale of the collection (in number and dimension) increases. While even young children have the ability to group objects, e.g. fruits, by size, color, or some other feature or combination of features, finding clusters among thousands of genes using measurements taken over hundreds of experiments is a task well beyond even the most analytical mind.

Several algorithmic tools have been developed to find clusters in data, each taking a different view of what constitutes a "good" cluster and how to find such clusters. The oldest and most widely used approach, hierarchical agglomerative, makes the assumption that the span between any pair of points determines the degree to which they should be in the same cluster, and finds a clustering by agglomerating points separated by minimum spans. Another widely used approach, k-means, assumes that clusters form dense balls in space and finds an appropriate assignment of data points to balls via an iterative reassignment algorithm.

Independent of their particular model assumptions and algorithmic details, these algorithms and the vast majority of clustering algorithms in common use make the assumption that every point must belong to one and only one cluster. This property, called "single-membership", simplifies algorithms and analysis, but may not represent the cluster structure that might be expected in several real world situations. The concept of a given object belonging to several clusters, or to none, is quite natural and is widely represented using the familiar Venn diagram, as shown on the left of Figure 1.1 contrasting with a strict single-membership assignment on the right. Clusterings that find solutions more like Venn Diagram are called "overlapping" or "multiple-membership".

Specific examples from several domains suggest that the multiple-membership representation may be closer to reality:

- Grouping documents into topic clusters - documents can contain multiple topics;
- Using purchasing histories to group customers into classes - customers often exhibit composite behaviors;
- Assigning movies into genre clusters based on viewer reviews - movies can be members of multiple genres;

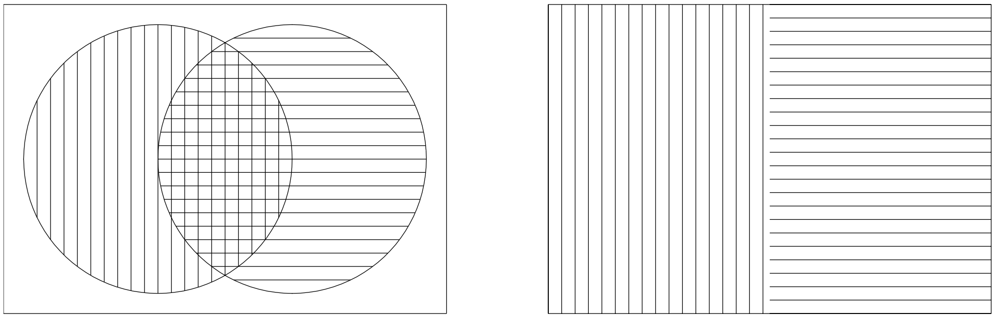

Figure 1.1: Venn diagram representing label assignments of points, and traditional clustering labelling of the same situation. Note that the Venn diagram allows points to be labelled as belonging to one, both, or no clusters, while the traditional clustering requires each point to belong to exactly one cluster.

- Determining biological process gene clusters based on large scale observations of gene regulation in response to environmental stimuli - individual genes are known to be required for several processes.

This dissertation describes the common features of this type of data along with methods for finding and evaluating multiple-membership clusterings.

1.1 Problem Statement of Clustering

Any clustering problem can be thought of as having an input space x and an output space y. The task of clustering is to find a mapping $y = C(x)$ whereby instances in x are mapped through C to instances in y in such a way as to impart an understanding of the data in x via the concise representations in y. In the language of clustering, the instances in x are called *data points* and the corresponding instances in y are called *labels*. The operator C is called the *clusterer*.

The task of the clusterer is to summarize and compress the potentially overwhelming (i.e. vast and high-dimensional) data in x into a human-consumable form in y. What constitutes the limits of "human-consumable" may be debatable, however it is generally accepted that labels that can be interpreted as answers to "yes/no" questions are easily understood. The dominant forms of clustering, notably k-means and hierarchical agglomerative, provide labels for data points which state clearly "this instance is a member of a particular group, and is not a member of any other group". The algorithms find these labels by minimizing objective functions that attempt to preserve as much detail about the original data, x, using the constrained output space, y.

This "preservation of detail" is quantified as the minimization of the distortion between x and y can be viewed as a problem of communication: we desire to send messages from the set y (the labels) that communicate as much information about x (the data points) as possible while conforming to the given constraints, e.g. that y be "human consumable". *Information* in this context has a specific meaning: it is the reduction in uncertainty we have about a point from x given its corresponding message from y. [1]

The constraints placed on y define how much information can be communicated by a single message. A complete relaxation of "interpretability" constraints on y would result in the message simply being the data point it-

---

[1] For example, consider a collection of children's building blocks where half the blocks are big, half small, half green, and half red. If we know that someone has drawn a particular block but we have not recieved a message about the block, we could not say anything confident about a given block beyond the prior probabilities: there is a 50% chance the block is big, 50% chance it is small, 50% it is green and 50% it is red. After seeing a label, e.g. "big, green", we can eliminate many possibilities for our observation - the block is neither small nor red. Our uncertainty about the observation has been reduced.

self. This scenario results in perfect communication of information - we have no uncertainty about the data point - but we have been given no advantage in interpretability. A slightly stricter relaxation of the constraints on y would be that the elements of y be real numbers (e.g., representing weights on basis vectors). This scenario yields SVD-like decompositions (PCA, Factor Analysis) which provide some interpretability at the cost of some (small) uncertainty in x. A very strict imposition of the interpretability constraint - that each vector in x be represented by a single label from y - yields standard single-membership clustering. Knowing a data point's label reduces our uncertainty inversely proportional to the number of data points sharing each label.

Multiple-membership clustering fits into the interpretability/distortion continuum somewhere between the SVD-like decompositions and single-membership clustering. Multiple labels per data point allow the communication of more information at the cost of the human consumer having to interpret multiple yes/no labels. While this may be slightly more challenging than interpreting a single yes/no label, is it surely less onerous than interpreting a vector of real numbers.

1.2 Notation

This section introduces the specific terminology and notation this paper uses in describing data, algorithms, and results.

Data is usually presented as a matrix X consisting of n rows where each row represents a data point and m columns where each column represents a feature of the data. Thus, the cell $X[i, j]$, represents the value of the jth feature of the ith data point. Each of the m features can be considered a "dimension" of the data, so a data set with m features (columns) can be said to be "m-dimensional". $X[i,:]$ represents the i^{th} m-dimensional row of X, and $X[:,j]$ represents the j^{th} n-dimensional column of X.

Standard disjoint clustering algorithms return an n-length vector of labels, giving each point one of k labels. An alternative view of this result is the *membership matrix*. A membership matrix M is a binary matrix with n rows, one for each data point, and k columns, one column for each label. If a data point i is a member of cluster j, then $M[i,j] = 1$. If i is not a member of j, then $M[i,j] = 0$. Traditional single-membership clustering algorithms return membership matrices where the the sum across any row will be 1, $\sum_{j=1}^{k} M[i,j] = 1$, indicating that each data point is assigned only one label. Membership matrices returned by multiple-membership clusterings do not have this constraint; instead $0 \leq \sum_{j=1}^{k} M[i,j] \leq k$, where each $M[i,j] \in 0,1$ and $\sum_{j=1}^{k} M[i,j]$ is an integer. *Soft* clustering algorithms return *soft* membership matrices, $M_{(\mathrm{soft})}$ where each $M_{(\mathrm{soft})}[i,j]$ specifies the fractional membership of data point i in cluster j and $\sum_{i=1}^{k} M_{(\mathrm{soft})}[i,j] = 1$.

Henceforth, clustering algorithms will be referred to as "clusterers" and will be given capital alphabetic labels such as C. The membership matrix returned by clusterer C will be labelled M_C.

1.3 Objectives, Contributions, and Organization

The central goal of this dissertation is to serve as a foundation for understanding of multiple-membership clustering. Towards that goal, it explores analysis and review of existing algorithms, new algorithms, and approaches to analyzing and effectively using multiple-membership clustering.

Despite the intuitive appeal of multiple-membership clustering, its exact definition and applicability needs to be clearly defined. Chapter 3 (Comparison) provides an intuition for clustering as a model of communication and extends that intuition to multiple-membership clustering and the types of problems where it is appropriate.

While single-membership methods are dominant in common practice, several multiple-membership techniques have been developed for clustering problems arising in diverse fields including psychology, sociology, economics, and biology. Given that most of these techniques arose as *ad hoc* solutions to domain-specific problems, they have not been widely publicized or employed outside of their original domains. Chapter 2 provides a survey of previous work in the field beginning with a discussion of k-means and hierarchical agglomerative as representatives of the model-based and non-model based approaches to single-membership clustering. Chapters 5 and 6 describe novel multiple-membership formulations of these two fundamental approaches. Chapter 4 addresses the particular multiple-membership problem of cluster alighnment and provides a numeric and visual means of multiple membership cluster comparison. Chapter 3 describes multiple-membership generalizations of single-membership cluster comparison metrics. Chapter 7 concludes and offers directions for future work.

Chapter 2

Background

While the idea of multiple-membership clustering has recently emerged as an important research topic as machine learning encounters problems arising from biology and other areas, the idea is not new. Since the 1960's, researchers in several fields, ranging from sociology and marketing to psychology and neuroscience, have encountered similarly messy problems and have developed multiple-membership clustering approaches for their domains. In this chapter, we review older as well as more recent multiple-membership clustering techniques and describe them via their commonalities with widely-used single-membership approaches.

2.1 Single Membership Clustering

Clustering is a form of *unsupervised* learning, meaning that no labels are available for training - labels must be deduced solely from the arrangements of the points in the input space. Thus, clustering algorithms are designed to find clusters using observable input space features such as distances between points and regional densities. While single-membership and multiple-membership clustering have different conceptions of the constitution of a "good" clustering, they both have as their input the same view of the data, and therefore employ similar approaches to extracting meaning from that input. These approaches can be described concisely as being either local, global, or some combination

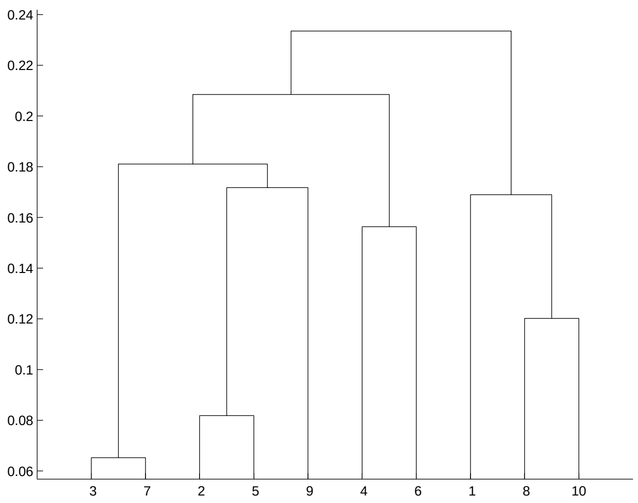

Figure 2.1

of the two. Examples from single membership clustering are instructive: the two most widely known and used algorithms - hierarchical agglomerative and k-means - are good representatives of local and global approaches, respectively.

Hierarchical agglomerative clustering, henceforth "HAC", is a local algorithm based on the idea that if two points are close together in the input space, they should be in the same cluster in the output space. At each step, the algorithm performs a local action: it finds the two closest points and merges them. The greedy nature of this local operation ensures that after several repetitions, the data will be grouped into clusters.[1] The sequence of merges results in a visual representation called a dendrogram, as shown in Figure 2.1.

The result of the merging process is not exactly a clustering on its

---

[1] There are several ways of defining "close" when using HAC - e.g., single-link, complete-link, mean-link. One of these, single-link, is the solution to the global minimum-spanning-tree problem[Har75]; however, it is not generally the case that local approaches solve global problems.

own; further refinement is necessary to select which of the nested clusters are most meaningful. Techniques for selecting clusters from hierarchical trees fall into two categories: those that choose to cut the tree at a particular level in order to return a particular number of clusters, and those that choose clusters individually according to some criteria (e.g. cluster density, stability, etc.).

In contrast to local algorithms like HAC, global model-based approaches assume that the distribution of points in the input space is due to sampling from an underlying latent probability model. The global problem is then to find the model parameters that maximize the likelihood of the observed points. The k-means algorithm [Mac67, Spa73, HW79] is an example of a global, model-based approach to clustering. The k-means model assumption is that the observed points are sampled from k spherical Gaussians. Once the means are known, each point is most likely to have been generated by the Gaussian centered on the closest mean (in squared-Euclidean sense). Finding the means is the parameter estimation problem, and is accomplished by a greedy iterative-reassignment algorithm:

1. Choose k centers at random.

2. Assign each point the label of its nearest center.

3. Move each center to the mean of the points with its label.

4. Repeat 2-3 to convergence.

HAC and k-means' different approaches - one looking exclusively at local, pairwise information and the other looking at some global objective -

represent the two fundamental patterns of unsupervised clustering. All unsupervised methods, single-membership and multiple-membership, exhibit characteristics of one or both of these approaches.

2.2 Multiple Membership Clustering

2.2.1 Soft Models

Soft models can be thought of as global clusterers which, instead of committing a point to be a member of a particular cluster, allow it to be a partial member of some or all clusters. The k-means algorithm described above performs hard assignment in the second step: each point is assigned the label of its nearest center. A soft version of k-means would instead assign each point a fraction of each label in proportion to the point's distance to each representative center. At the conclusion of the algorithm, the resulting assignment for each point would, instead of being a particular label, be a set of fractional labels. Thresholding the fractional labels so that all greater than the threshold are assigned to the point and all less than discarded yields, in effect, a multiple-membership assignment of labels.

There are two primary methods of finding soft clustering representations of data. The soft k-means approach described previously is a representative of EM-like models - iterative reassignment algorithms that assume the observed data to be a mixture of several *mixture components*, each a random variable/vector with an associated distribution. The other type of soft approach is based on SVD-like matrix decompositions.

2.2.1.1 SVD-like matrix decompositions

As mentioned in Chapter 1, if single-membership clustering represents the maximum interpretability end of the interpretability/distortion spectrum then SVD-like decompositions represent the minimum distortion end. Matrix decompositions attempt to find simpler representations of observed matrices that preserve most of the information contained in the original matrix. Generally, matrix decompositions used in data mining applications follow the general model of *Factor analysis*. The factor analysis model assumes every observed row (feature vector) to be a linear combination of global factors (potentially affecting all points) and local factors (only affecting a particular point). Factor analysis was originally developed as a tool for psychometrics where, for example, the global factors could be mental characteristics (e.g. mathematical aptitude, linguistic aptitude, etc.) and the local factors could be conditions specific to the administration of a given test. A group of subjects' test performance would then be decomposed by factor analysis into their global features (aptitudes) and local features (test conditions). In more recent application of factor analysis to data mining problems, local features are largely ignored, making SVD-like decompositions the preferred tool.

Factor analysis decompositions that find global factors from SVD-like decompositions are not clusterings *per se*, but can be converted to clusterings by thresholding. Depending on threshold choice, these resultant clusterings can be multiple-membership.

The *Singular Value Decomposition* (SVD)[GL96] performs the eigendecomposition of the data matrix X into an orthogonal column basis U, an orthogonal row basis V, and a diagonal matrix of eigenvalues, Σ:

$$X = U\Sigma V^T \tag{2.1}$$

The columns of U are the eigenvectors of the column space and the rows of V are the eigenvectors of the row space. When the primary interest is in the rows, U and Σ can be combined into a weight matrix W yielding $X = WV^T$. This transformation is useful because it represents every point in X as a linear combination of the rows of V^T:

$$X[i,:] = \sum_{j=1}^{k} W[i,j]V^T[i,:] \tag{2.2}$$

If each row of V^T is thought of as a cluster center, then each row of W defines the soft cluster membership of the corresponding row in X. The application of a threshold to W yields a multiple-membership clustering membership matrix. When correlation is more relevant than the additive coefficient, a similar weight matrix can be constructed such that $W[i,j]$ contains the correlation coefficient between data row i and eigenrow j. As with the additive coefficient weight matrix, thresholding this matrix yields a multiple-membership clustering membership matrix. Choosing the threshold in either case is a domain-specific task which requires some labeled data or at least knowledge of the expected cluster priors. Alter *et al.* [ABB00] has shown very good results using SVD with correlation thresholding on yeast cell cycle data.

The soft form of SVD-based clustering is widely used in the Latent Semantic Indexing (LSI) [DDL$^+$90, FDD$^+$88] of text corpi. In LSI, the rows of V^T (the eigenvectors of the term-frequency data matrix) are interpreted to be representations of the latent topics of the document corpus, and the values of the W matrix as each document's expression of those latent topics.

The LSI approach to text clustering performs well, but has a shortcoming in interpretability. The SVD decomposition places no constraints on

V^T and thus, V^T and W both contain positive and negative values. LSI topic vectors, then, contain positive and negative numbers corresponding to the expected distribution of word counts in documents from that topic. The concept of negative word counts is problematic.

To provide a more interpretable decomposition, Lee and Seung proposed non-negative matrix factorization (NMF) [LS00]. NMF performs the SVD-like decomposition of a non-negative data matrix X into a non-negative weight matrix W and non-negative basis matrix H:

$$X = WH \tag{2.3}$$

NMF has been shown in document clustering [XLG03] and other fields such as bioinformatics [BTGM04] and image processing [LS99] to give highly interpretable decompositions. Just as with SVD, the rows of the H matrix can be interpreted as cluster centers and the rows of W as soft memberships, with a multiple-membership clustering arising from thresholding W.

Certain other decompositions such as Independent Components Analysis (ICA) can similarly be interpreted as soft clusterings and can be mapped to multiple-membership clusterings by thresholding.

2.2.1.2 Gene Shaving

The appropriate choice of thresholds for soft methods like the SVD can be difficult. The “Gene Shaving” algorithm [HTE$^+$00] employs a statistical notion of the meaning of a cluster in order to eliminate the need to choose thresholds.

The algorithm examines each eigenrow (potential cluster center) in order with a two phase process. In the first phase, the algorithm generates a series of increasingly compact nested clusters centered on the eigenrow. The second phase computes for each cluster in the nested series the “gap statistic”, a measurement of the extent to which the cluster minimizes the variance between its members and maximizes the difference between its mean and the global mean of the data. The cluster with the highest gap statistic is taken as the cluster for the current eigenrow, the data is orthogonalized with respect to the cluster mean, and the process repeats. The algorithm continues until a set number of clusters is produced, or the residual data after orthogonalization is noise.

2.2.2 Multiple-Membership Extensions to Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering (HAC) is a conceptually simple clustering algorithm and has served as the starting point for several multiple membership clustering algorithms.

2.2.2.1 Jardine-Sibson B-clustering and Articulation Point Cuts

Jardine and Sibson’s “B-clustering”[JS68] is a straightforward extension of single-link agglomerative clustering. On each step of the standard single-link HAC algorithm, the most similar pair of points and the clusters containing them are merged. Therefore, two clusters that share little similarity on the majority of their points can be joined if their fringe points are in close proximity. This property leads to “chaining” - the propensity for forming long, spindly clusters that is the commonly-cited flaw in single-link HAC. B-clustering solves this problem by allowing points to be merged into their closest cluster, but requiring clusters to share *at least* b points before they are

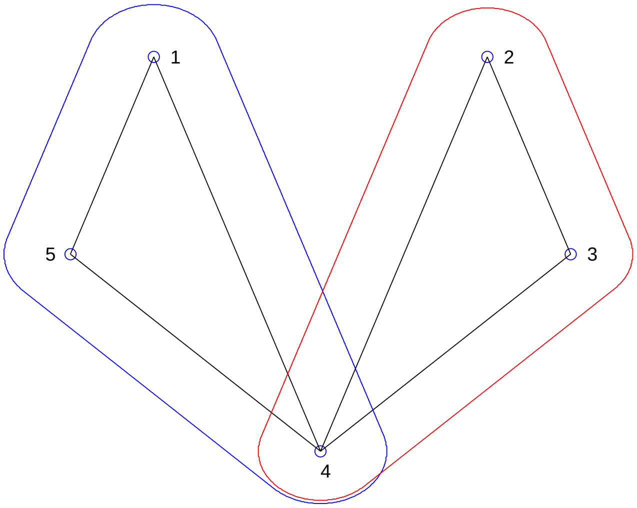

Figure 2.2: Jardine-Sibson cluster for $b = 2$

merged. Thus, if $b = 2$, clusters A and B can share a single point x, yet remain distinct clusters with x simultaneously a member of both A and B. Figure 2.2 illustrates Jardine-Sibson clustering for $b = 2$.

A closely related approach to B-clustering is the problem of finding *articulation vertices* in graphs [GI91, Har69]. An *articulation vertex* is a node in a graph that has high connectivity to two or more dense subgraphs. Instead of associating the articulation vertex with one of the dense subgraphs - which would require an expensive cut - the articulation vertex can be copied into all of the subgraphs, effectively associating it with multiple clusters. Bejerano *et al.* demonstrated a bioinformatics application of this approach in [BHB04].

2.2.2.2 Pyramid Hierarchical Clustering

In [BD85], Bertrand and Diday propose an alternate overlapping extension to HAC which can be thought of as a relaxed form of HAC. The HAC algorithm maps points in a metric on non-metric space into the closest (i.e. minimum distortion) ultrametric space, where there is a unique ordering which can be represented as a dendrogram. Pairwise distance matrices of ultrametric embeddings (henceforth, ultrametric matrices) have the properties that values are symmetric, non-decreasing away from the diagonal and that for any two points a and b, there exists a point c such that $d(a,b) \leq max[d(a,c), d(b,c)]$. Relaxing the second constraint to the triangle inequality, $d(a,b) \leq d(a,c) + d(b,c)$, gives Robinsonian matrices, where values are symmetric and non-decreasing away from the diagonal. Robinsonian matrices are a superset of ultrametric matrices and do not in general guarantee a unique ordering, but instead a set of nonconflicting or "compatible" orderings. Bertrand and Diday's approach find the closest (minimum distortion) Robinsonian embedding for a given set of similarities.

The existence of several compatible orderings gives rise to overlapping clustering by allowing points (or groups of points) to belong to at most two different clusters, with both possible assignments allowing compatible orderings. The maximum membership of two arises from the fact that orderings are 1-dimensional; a point has a left (less-than) side and a right (greater-than) side, and can at most belong to clusters on both its left and right.

The pyramid approach has not found widespread use, however, it was used successfully by Aude *et al.* in [ADLCJ99] to find an overlapping clustering of selected genes from 5 different organisms (*E. coli, S. cerevisiae, M. Janaschii, H. influenza, and Synechocystis*). Genes assigned to two clusters generally contained protein domains common to both, while genes assigned to single clusters had only one type of domain. Figure 2.3 show an example pyramid from Aude *et al.*'s data.

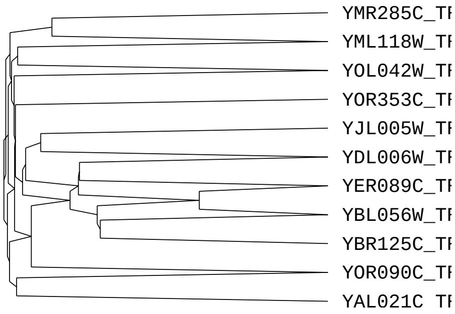

Figure 2.3: Pyramid hierarchical clustering of yeast genes from Aude, *et al.* [ADLCJ99]

2.2.3 Similarity-Space Additive Clusterings

In [AS73, Ara77], Phipps Arabie proposed an additive model called "ADCLUS" for modelling similarity matrices. An observed similarity between two objects, he posited, could be considered to be a weighted sum of several subsimilarities between particular properties of the objects. Mathematically, the similarity s between the objects i and j is then:

$$s_{i,j} = \sum_{k=1}^{m} w_k f_{i,k} f_{j,k} \text{ where } f_{i,k} = \begin{cases} 1 & \text{if object } i \text{ has property } k \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

By considering the presence or absence of each feature $f_{i,k}$ to be a cluster label, each point is assigned k binary labels indicating its membership in the k clusters.

The ADCLUS model is compelling for several reasons. First, it is conceptually simple and intuitive. Second, it requires only a similarity matrix, meaning that the model is not affected by the actual input space features (e.g. nomative, ordered, etc.) as long as similarity between points is defined. Finally, it provides a weight for each cluster which is convenient for interpretation and discarding unimportant clusters.

When stated in matrix notation, the model's relation to SVD-like decompositions is clear:

$$\hat{S} = FWF^T$$

where S is the $n \times n$ similarity matrix, W is an $m \times m$ diagonal matrix of weights, and F is a binary $n \times m$ property membership matrix. Without the requirement that F be binary, this is exactly the SVD of the similarity matrix.

Analogous to SVD, fitting the ADCLUS model requires minimizing the squared-error energy function:

$$E = \sum_{i \neq j} (s_{i,j} - \hat{s}_{i,j})^2 = \sum_{i \neq j} (s_{i,j} - \sum_k w_k f_{i,k} f_{j,k})^2 \tag{2.5}$$

If f were real instead of binary, the energy function could be minimized by alternating least squares minimization of f and w; however, the binary constraint on f does not permit that approach. Instead, f must be minimized combinatorially by exploring all 2^k binary vectors. This is clearly not acceptable, and algorithms for fitting ADCLUS focus on reducing the complexity of this step. Once the optimal binary assignment of clusters is known, w can be optimized by linear least-squares.

Arabie and Shepard's "ADCLUS" algorithm and Arabie and Carroll's "MAPCLUS" algorithm both employed complicated *ad hoc* heuristic combinatorial optimization approaches to fit the model, resulting, according to Tenebaum in "sometimes ... unexpected or uninterpretable final results" [**?**]. Both algorithms follow the pattern of use of hierarchical clustering, that is, they first enumerate a large set of potential clusters, and then refine the set into a small number of meaningful clusters. The large set of potential clusters is taken to be all *elevated subsets* in the data, *elevated subsets* being dense sets of points where any additional points or sets of points will reduce the minimum similarity of the set.[2] The set of all elevated subsets is, like the set of all nested clustering from HAC, too large to be interpretable. The alternating stages of ADCLUS and MAPCLUS refine the large number of subsets by, through alternating optimizing assignments and weights, winnowing away those that are not required ($w = 0$) for minimizing the ADCLUS energy given in Equation 2.5.

Despite the algorithmic complications, ADCLUS and MAPCLUS have been demonstrated to be effective at finding overlapping clusterings in several fields, including social networks [Ara77], psychological tests [SA79], and retail product positioning [ACDW81]. It is notable, however, that the datasets in all cases are small - less than 30 points; neither the process of finding elevated subsets nor the combinatoric optimization scale to larger datasets. Even so, the appeal of the additive similarity model has led to several additional approaches to fitting it.

Hojo [Hoj83] proposed a maximum likelihood formulation called "MLADCLUS" which employs a likelihood function that returns the likelihood of a set of property assignments producing the observed ordering of pairwise similarities. Hojo's likelihood function allows model selection by Akaike information criteria, as well as goodness of fit computation by χ^2 tests.

Tenenbaum reformulated the model-fitting problem as an instance of expectation-maximization [Tan96], providing a clearer view of the optimization process. Tanenbaum observed that the probability of a set of observed similarities, s, for a given cluster assignment f and weighting w is:

---

[2] Finding elevated subsets can be viewed as a multiple-membership generalization of complete-link hierarchical clustering, and is detailed in Chapter 6.

$$p(s|f,w) \propto \exp\{-\frac{1}{2\sigma^2}E\} = \exp\{-\frac{1}{2\sigma^2}\sum_{i\neq j}(s_{i,j} - \sum_k w_k f_{i,k} f_{j,k})^2\} \qquad (2.6)$$

The E-step of EM is then:

$$Q(w|w^{(n)}) = \sum_{f'} p(f'|s,w^n) \log p(s,f'|w) = \frac{1}{2\sigma^2} < -E >_{s,w^n}$$

which requires averaging E over all possible configurations of f, the cluster assignments. Tenenbaum ameliorates this combinatoric problem by using annealing and Gibbs sampling to estimate the expected value of E. The results are identical to those of the ADCLUS and MAPCLUS algorithms on letter-recognition data; however, even with the speedups in estimating E, this algorithm becomes unmanageable for large (i.e. >30) numbers of points or clusters.

More recently, Navarro and Griffiths in [NG05] described fitting the ADCLUS model in a Bayesian framework. They stated the ADCLUS likelihood function as:

$$p(S|F,w,\sigma) = \prod_{i<j} \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{1}{2\sigma}(s_{i,j} - \sum_k w_k f_{ik} f_{jk})^2) \qquad (2.7)$$

which differs from Equation 2.6 in the inclusion of the normalization term ($\frac{1}{\sigma\sqrt{2\pi}}$) and in evaluating only the upper triangle of the similarity matrix ($i < j$). The two formulations are identical to a constant scalar.

Navarro and Griffith's Bayesian approach defines priors for the variables to be determined, F and w. The prior on w is a simple exponential, $p(w_k) =$

$\frac{e^{-\lambda w_k}}{\lambda}$, which serves as a regularizer for the w_k's. The prior on the cluster memberships, F, is provided by the *Indian Buffet Process* [GG05], a generative process which, as the Chinese Restaurant Process does for single-membership clusterings, provides sample instantiations of multiple-membership clusterings. The complete model is then:

$$\begin{array}{rcll} s_{ij} & \mid & F, w, \sigma & \sim \text{ Normal} \\ w_k & \mid & \lambda & \sim \text{ Exponential} \\ F & \mid & \alpha & \sim \text{ Indian Buffet Process} \end{array}$$

Navarro and Griffith employ Gibbs sampling to compute the conditional posterior on feature assignments. Using the IBP prior and Gibbs sampling instead of a combinatoric approach gives a tremendous speed advantage, allowing this method to fit the ADCLUS model on datasets consisting of hundreds of points.

2.2.3.1 Decomposing Similarity Graphs: MODES

In [HYH$^+$05], Hu *et al.* describe an iterative algorithm called "MODES" (**M**ining **O**verlapping **DE**nse **S**ubgraphs) for finding overlapping dense subgraphs in graphs. MODES finds overlapping dense subgraphs by iterative application of a modified version of the (nonoverlapping) dense subgraph finding algorithm "Highly Connected Subgraphs" (HCS) [HS00]. HCS identifies dense subgraphs by recursively partitioning a graph by mincuts and checking each resultant subgraph for complete connectivity. MODES' modified HCS uses normalized cut [SM] to overcome HCS's tendency to make highly unbalanced cuts. On each outer iteration, MODES identifies connected subgraphs by normalized cut HCS, and then condenses the discovered subgraphs into single vertices that maintain all of the contained vertices' links to external vertices. On subsequent outer iterations, if previously condensed vertices identified as part of a dense subgraph, they are expanded and their contained point incorporated into the new dense subgraph.

2.2.4 Feature Space Additive Clustering

Similarity space additive clustering assumes that the observed similarity between data points is due to an additive combination of the similarities due to the points' unseen individual properties. A benefit of this model is that similarity between points is always available or can be computed. However, in situations where data features are observable, using a similarity matrix may obscure meaning in the features themselves. Feature space additive models address this by considering the observed features themselves to be an additive combination of unseen latent features. Thus, a given input space vector X would be represented as

$$X[i,:] = m_i^T \mathbf{F} \tag{2.8}$$

where m_i is a k-length binary vector of memberships and $\mathbf{F}$ is a $k \times d$ matrix with latent factor vectors as its rows. An observation $X[i,:]$ is then a linear combination of factors. The whole dataset $\mathbf{X}$ would be:

$$\mathbf{X} = \mathbf{MF} \tag{2.9}$$

where each column of $\mathbf{M}$

Then, the jth feature of the ith data point would be:

$$X_{ij} = \theta_{ij0} + \sum_{k=1}^{K} \theta_{ijk} \rho_{ik} \kappa_{jk} \tag{2.10}$$

where X_{ij} is the observed data point, θ_{ij0} is the background activation level of point (i, j), θ_{ijk} is the activation of layer (process) k at point (i, j), ρ_{ik} is a binary indicator of row i's membership in layer k, and κ_{jk} is a binary indicator of column j's membership in layer k. A non-overlapping model would require that $\sum_k \rho_{ik} = 1$ and $\sum_k \kappa_{jk} = 1$, but the Plaid model explicitly allows $\sum_k \rho_{ik}$ and $\sum_k \kappa_{jk}$ to take any nonnegative integer value.

2.2.4.1 The "Plaid" Model

Lazzeroni and Owen proposed the Plaid model in 2000 for decomposing microarray data into layers, each layer corresponding to a clustering of genes and conditions. (The name "plaid" comes from the intersecting colored stripe appearance of the clusters.) The Plaid model is very similar to Shepard and Arabie's ADCLUS, except that ADCLUS decomposes a similarity matrix while Plaid decomposes a data matrix. Each element of the input matrix is modeled as a sum of activations,

$$X_{ij} = \theta_{ij0} + \sum_{k=1}^{K} \theta_{ijk} \rho_{ik} \kappa_{jk} \tag{2.11}$$

where X_{ij} is the observed data point, θ_{ij0} is the background activation level of point (i, j), θ_{ijk} is the activation of layer (process) k at point (i, j), ρ_{ik} is a binary indicator of row i's membership in layer k, and κ_{jk} is a binary indicator of column j's membership in layer k. A non-overlapping model would require that $\sum_k \rho_{ik} = 1$ and $\sum_k \kappa_{jk} = 1$, but the Plaid model explicitly allows $\sum_k \rho_{ik}$ and $\sum_k \kappa_{jk}$ to take any nonnegative integer value.

The model attempts to minimize the total squared reconstruction error:

$$\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{p} \left(Y_{ij} - \theta_{ij0} - \sum_{k=1}^{K} \theta_{ijk} \rho_{ij} \kappa_{jk} \right)^2 \tag{2.12}$$

This minimization is NP-hard due to the binary constraint on ρ and κ. As with ADCLUS, Plaid attempts to find an optimal additive model for k layers by first greedily adding layers and then minimizing it's objective by iteratively cycling through minimization of θ, and real-relaxed minimizations of ρ and κ.

Lazzeroni and Owen demonstrate Plaid on three datasets, concluding that it successfully gives interpretable, overlapping clusters.

2.2.4.2 The SBK Model: Decomposing Gene Expression into Cellular Processes

In [SBK03a], Segal, Battle and Koller proposed a probabilistic model of a microarray dataset. This model, henceforth "SBK", models each observed expression value as a sample drawn from a Gaussian whose mean is a sum of real-valued activations of processes that a gene participates in. Explicitly, where:

- The input data, X, is a real-valued $(n \times m)$ matrix where n is the number of genes and m is the number of experimental conditions
- M is a $(n \times k)$ binary membership matrix
- A is a $(k \times m)$ real-valued activity matrix

$$P(X[i,j]) = exp\left(\frac{(X[i,j] - M[i,:]A[:,j])^2}{2\sigma_j^2}\right) \tag{2.13}$$

and thus,

$$E[X[i,j]] = M[i,:]A[:,j] \tag{2.14}$$

The problem then is to find M and A so as to maximize the joint probability,

$$\begin{aligned} p(X, M, A) &= p(M, A)p(X|M, A) = p(M)p(A)p(X|M, A) \\ & \left(\prod_{i,h} p(M[i, h])\right)\left(\prod_{h,j} p(A[h, j])\right)\left(\prod_{i,j} p(X[i, j]|M[i, :], A[:, j])\right) \end{aligned}$$

With the assumption that the $A[j, h]$ are uniformly distributed and that the conditional distribution of X[i,j] is Gaussian, the maximization of the log-likelihood is then:

$$\begin{aligned} \max_{M,A} \log p(X, M, A) &= \max_{M,A} \left[\sum_{i,j} \log p(M[i, h]) - \frac{1}{2\sigma^2}\sum_{i,j}(X[i, j] - M[i, :]A[:, j])^2\right] \\ & \max_{M,A} \left[\frac{1}{2\sigma^2}\|X - MA\|^2 - \log p(M)\right] \end{aligned}$$

The core optimization is then to find M and A that minimize $\|X - MA\|^2$. The SBK model estimates M and A as follows:

1. M is seeded with a first estimate of the clustering in the data, usually the output of a partitional clustering such as hierarchical or k-means run on the rows of X.
2. Next, the least-squares approximation of A for the given X and M is found as $A = M^{\dagger}X$, where $M^{\dagger}$ is the pseudo-inverse of M.
3. Using the A from step 2, the next approximation of M is found by relaxing the requirement that M be binary and solving a bounded least squares optimization for each gene in M. This effectively seeks a solution $\hat{M}[i, :] = [0, 1]^k$ for each row such that $||X[i, :] - \hat{M}[i, :]A||^2$ is minimized.

4. A binary solution M is then recovered from the real-valued solution $\hat{M}$ found in step 3 by thresholding. Since thresholding potentially moves the solution away from optimal, a local search is performed over every possible 0-flip of the post-threshold 1's to find the $M[i,:] = \{0,1\}^k$ that minimizes $||X[i,:] - M[i,:]A||^2$.

5. Using the new M calculated in step 4, steps 2-4 are repeated until $||X - MA||^2$ is less than the desired convergence criteria.

The paper [SBK03a] demonstrates the application of this algorithm on the Gasch yeast stress response dataset [GSK$^+$00], finding that discovered overlapping clusters had much better label enrichment (as determined by hypergeometric p-value) than clusters discovered by the overlapping Plaid algorithm or non-overlapping HAC.

2.2.4.3 Biclustering

In [CC00], Cheng and Church give a biclustering (a.k.a. co-clustering) algorithm for finding biclusters in microarray data. A bicluster is a submatrix (rows I and columns J) that minimizes some objective. Cheng and Church find biclusters that minimize *mean square residue*:

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \tag{2.15}$$

where

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij}, \quad a_{Ij} = \tfrac{1}{|I|} \textstyle\sum_{i \in I} a_{ij}, \quad a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij} \tag{2.16}$$

The paper describes several greedy algorithms for finding single biclusters that minimize mean square residue, and an iterative algorithm for finding k biclusters which involves iteratively finding and masking (by random numbers) discovered biclusters. The iterative algorithm results in a coupled set of membership matrices, M_{row} and M_{col} where $M_{row}[i,:]$ defines the row indices of bicluster i, and $M_{col}[i,:]$ defines the column indices of bicluster i. Taken individually, M_{row} will contain an overlapping clustering of the rows and M_{col} will contain an overlapping clustering of the columns.

Cheng and Church demonstrate the biclustering algorithm on Alizadeh *et al.*'s human lymphoma dataset [ea00], showing qualitatively good results.

2.2.4.4 Bond-Energy and Rank-Order Clustering

The ADCLUS model can be thought of as representing an observed similarity matrix as a weighted sum of similarity matrices, each representing the similarity structure due to a particular feature. A less delicate approach would be to group together those points that exhibit high similarity, and to separates those points that exhibit low similarity.

The Bond Energy Algorithm (BEA) works on non-negative matrices and attempts to maximize sum of the "bond energy" between each entry and its immediate neighbors:

$$\sum_{i=1}^{M}\sum_{j=1}^{N} a_{i,j}\left[a_{i,j-1} + a_{i,j+1} + a_{i-1,j} + a_{i+1,j}\right]$$

(where $a_{0,j} = a_{M+1,j} = a_{i,0} = a_{i,N+1} = 0$). According to McCormick et al. [MSW72], an $O(N^4)$ quadratic assignment algorithm exists for finding the global optimum; however this is computationally prohibitive, and a greedy *best-insertion* algorithm is proposed as an adequate approximation (though still $O(M^2N+N^2M)$). The Rank-Order clustering algorithm (ROC) works on binary matrices by simply treating each row and columns as a binary number and alternatively numerically sorting the rows and columns until sorting does not change either order. Final cluster labelling in both algorithms is left to visual examination of the reordered matrix. Both algorithms permit overlap, but leave the difficult choice of thresholds and labelling to humans.

These matrix-reordering algorithms can be viewed as heuristic solutions to the problem of finding dense subgraphs [FPK01, GKT05] of a graph where each row and each column corresponds to a vertex, and each matrix entry corresponds to the weight of the arc connecting the row and column vertices. Blocks in the reordered matrices correspond to dense subgraphs, and overlapping points correspond to *articulation vertices* of the graph.

Chapter 3

Comparison of Overlapping Clusterings

An essential problem in both single-membership and multiple-membership clustering is evaluation and comparison of the results provided by different clusters. One of the appeals of single-membership clustering is that the notion of membership is simple: two points are either in the same cluster or they are not. Multiple-membership clustering requires a much more nuanced view: two points could be in exactly the same set of clusters, have no cluster memberships in common, or share a subset. This chapter explores how to reasonably compare two multi-membership clusterings, and derives some insights about the interpretation of overlapping clustering results.

Issues in comparing clusterings

In general, two clusterings of the same data will not use the same labels, or even the same number of labels (clusters). While the actual labels placed on the points are arbitrary, the distributional characteristics of the label assignments can be compared between the two clusterings. In particular, if the two clusterings are similar but with different labels, the knowledge of a point's labels in the first clustering should allow one to guess the point's corresponding label in the second clustering with an accuracy better than random [1]. Alternatively, the knowledge that two points share a label in the

---

[1] That is, better than a random draw from the prior distribution over labels for the second clustering.

first clustering should provide an improved guess about the likelihood that the same two points share a label in the second clustering. These two intuitions lead to the most commonly used measures for comparing disjoint clusterings: *mutual information*, *pairwise mutual information*, and the *Rand index*.

Evaluation and comparison of clustering results should be independent of the clustering algorithm generating the labels. The generated labels can be thought of as being samples from a random variable, providing a probabilistic model of labels generated by any particular algorithm.

This chapter is divided into three major sections. In the first, we describe a probabilistic model of single-membership clustering and deduce the comparison measures related to the intuitions described above. In the second section, we describe a probabilistic model of multiple-membership clustering and again deduce the comparison measures with special attention to the differences between single- and multiple- membership clustering. Finally, in the third section, we derive the property of label independence in multiple-membershp clustering.

3.1 Single-Membership Evaluation

3.1.1 Desiderata

The two desiderata we have for comparing single membership clusterings are:

(a) If two single-membership clusterings are similar, knowledge of the label assigned to a point by C_1 should reduce the uncertainty about the label assigned to the point by C_2.

(b) If two single-membership clusterings are similar, knowledge that two points are assigned the same label by C_1 should reduce the uncertainty about whether the pair is assigned the same label by C_2.

3.1.2 Probability Model of Single Membership Clustering

A single-membership clusterer assigns a label to each input point. If there are k labels, each label can be represented by an indicator vector:

$$
\begin{array}{c ccccc}
\text{Label} & \multicolumn{5}{c}{\text{Indicator Vector}} \\
L_1 & 1 & 0 & ... & 0 & 0 \\
L_2 & 0 & 1 & ... & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
L_{k-1} & 0 & 0 & ... & 1 & 0 \\
L_k & 0 & 0 & ... & 0 & 1
\end{array}
$$

A particular labeling returns a membership matrix, M:

$$
\begin{array}{ccc}
\text{points} & \text{labels} & \text{M (array of indicator vectors)} \\
\begin{array}{c} x[1] \\ x[2] \\ x[3] \\ x[4] \\ \vdots \\ x[n-2] \\ x[n-1] \\ x[n] \end{array} &
\begin{bmatrix} L_2 \\ L_4 \\ L_1 \\ L_2 \\ \vdots \\ L_3 \\ L_1 \\ L_4 \end{bmatrix} &
\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{array}
$$

where each row contains exactly one “1”.

If we know nothing about the points x, we can compute the prior probability of a particular point’s cluster assignment as:

$$p(\text{label}_i(x[m]) = 1) = \rho_i = \frac{\sum_{j=1}^{n} M[j, i]}{n}$$

The quantity ρ_i is the prior for cluster i, indicating the probability of a randomly chosen point having label_i.

We would expect, however, that a "good" clusterer would assign labels as a (possibly stochastic) function of the features of a data point, i.e.:

$$f_i(\text{features}(x[m])) = p(\text{label}_i(x[m] = 1|\,\text{features}(x[m]))$$

where $\sum_{i=1}^{k} f_i() = 1$.

3.1.3 Mutual Information

The first intuition mentioned in section 3.1.1 is that if two clusterings described similar clusters, then the knowledge of a point's label in the first clustering would be predictive of the point's label from the second clustering. This intution is measured mathematically by the *mutual information* between the cluster labels. The mutual information between two variables measures the Kullback-Leibler divergence ("KL divergence") between a system's observed state (the joint probability between the two sets of labels) and its maximum entropy state (the product of the priors of the two sets of labels) [CT91]. When used to evaluate the similarity between two clusterings A and B containing k and l unique lables respectively:

$$\begin{aligned} I(A;B) &= \sum_{i=1}^{k}\sum_{j=1}^{l} p(i,j)[\log(p(i,j)) - \log(p(i)p(j))] && (3.1) \\ &= \sum_{i=1}^{k}\sum_{j=1}^{l} p(i,j)\log\frac{p(i,j)}{p(i)p(j)} && (3.2) \end{aligned}$$

Inspection of the log ratio indicates that this quantity is zero when the joint distribution provides no information, $p(i,j) = p(i)p(j)$, and is positive when the joint diverges from the product of the priors. A problem with this form of the mutual information is that its range is dependent on the number of labels in sets A and B. Thus, this form does not allow one to answer the question "which clustering, B or C, is more similar to A?" if B and C have different numbers of clusters. To facilitate comparison, several normalized versions of the mutual information have been proposed which provide a value between 0 and 1 across different numbers of clusters. One of these proposed by Yao [Yao03] is:

$$\mathrm{NMI}(A;B) = \frac{\sum_{i \in A} \sum_{j \in B} p(i,j) \frac{\log(p(i,j))}{\log(p(i)p(j))}}{H(A,B)} \tag{3.3}$$

where

$$H(A) = -\sum_{i=1}^{k} p(i) \log_2 p(i) \tag{3.4}$$

The denominator arises from the expression of the mutual information in terms of entropies:

$$\mathrm{MI}(A;B) = H(A) + H(B) - H(A,B) \tag{3.5}$$

If the label assignments in A and B are completely independent, $H(A) + H(B) = H(A,B)$ and the mutual information is zero. If the label assignments in A and B are completely dependent, $H(A) = H(B) = H(A,B)$ and $H(A) + H(B) - H(A,B) = H(A,B)$. Intervening levels of dependence between A and B return values between 0 and $H(A,B)$, so normalizing by $H(A,B)$ keeps $\mathrm{MI}(A;B)$ between 0 and 1.

In the terms defined in section 3.1.2, the joint probability is $p(\text{label}(x[m]), \text{label}(x[m]))$ and the product of the priors is $p(\text{label}(x[m]))p(\text{label}(x[m]))$. The mutual information is then:

$$\sum_{i=1}^{k}\sum_{j=1}^{k} p(\text{label}_i(x[m]), \text{label}_j(x[m])) log_2 \frac{p(\text{label}_i(x[m]), \text{label}_j(x[m]))}{p(\text{label}_i(x[m]))p(\text{label}_j(x[m]))}$$

3.1.4 Pairwise Mutual Information

The second intution mentioned in section 3.1.1 was that if two clusterers found similar clusters, then a pair of points assigned the same label by the first clusterer would likely also be assigned the same label by the second clusterer. This intution is measured mathematically by the *pairwise mutual information* (PMI). The PMI measures the KL divergence between the joint probability that two different clusterers will assign a pair of points the same label and the product of the priors that each clusterer will assign points to the same label. Since in single-membership clustering, two points can either share a label or not share a label, the PMI is a measure of the mutual information between two Bernoulli random variables.

When comparing two Bernoulli RV's, only four things can happen, as shown in Table 3.1.5. The diagonal terms, a and d correspond to 11 and 00, respectively, and the off-diagonal terms, b and c correspond to 10 and 01, respectively. Using these terms, the PMI can be calculated as:

$$\begin{aligned}\text{PMI}(C_1, C_2) \quad = \quad & \frac{a}{N}\log_2\frac{a}{N} - \frac{a}{N}\log_2[(\frac{a+b}{N})(\frac{a+c}{N})] + \frac{b}{N}\log_2\frac{b}{N} - \frac{b}{N}\log_2[(\frac{a+b}{N})(\frac{b+d}{N})] \\ & + \frac{c}{N}\log_2\frac{c}{N} - \frac{c}{N}\log_2[(\frac{a+c}{N})(\frac{c+d}{N})] + \frac{d}{N}\log_2\frac{d}{N} - \frac{d}{N}\log_2[(\frac{c+d}{N})(\frac{b+d}{N})]\end{aligned}$$

	Pairs Together B	Pairs Apart B	
Pairs Together A	a	b	a+b
Pairs Apart A	c	d	c+d
	a+c	b+d	a+c+b+c = N

Table 3.1: Terms used in PNMI and Rand Index calculations

As with the mutual information, the pairwise-mutual information can be normalized by the joint entropy of the two random variables:

$$H(C_1, C_2) = -\frac{a}{N}\log_2(\frac{a}{N}) - \frac{d}{N}\log_2(\frac{d}{N})$$

yielding

$$\text{PNMI}(C_1, C_2) = \frac{PMI(C_1, C_2)}{H(C_1, C_2)}$$

3.1.5 The Rand Index and Adjusted Rand Index

While the PNMI is soundly rooted in information theory, its calculation can be cumbersome. The much simpler Rand Index [Ran71] is probably the most widely used method for comparing two different clusterings. As with the PNMI, the procedure for calculating the Rand index involves first tabulating the number of pairs of points with the same label in both clusterings, the same label by one and not the other, and the same label in neither (Table 3.1.5).

From this table, the Rand Index is:

$$R = \frac{a+d}{N} = \frac{\text{number of matching pairs}}{\text{total pairs}}$$

The Rand Index returns a number between 0 and 1 indicating the similarity between the two clusterings. However, if clusterers A and B randomly assign points to clusters, the products of the priors will yield some points sharing the same label by chance yielding a nonzero Rand index. The expected number of points together by chance would be:

$$E[R] = \frac{(a+b)(a+c)}{N^2} + \frac{(c+d)(b+d)}{N^2} \tag{3.6}$$

Removing these chance occurences yields the *Adjusted Rand Index* (ARI) [HA85]:

$$R_{adj} = \frac{\frac{(a+d)}{N} - E[R]}{1 - E[R]} = \frac{\text{observed index} - \text{expected index}}{\text{maximum index} - \text{expected index}}$$

The ARI approaches 1 as the two clusterings identically assign pairs, and approaches 0 as the two clusterings assign pairs the same label at a rate no greater than chance.

3.1.6 ARI and PNMI

The ARI and the PNMI are both computed over pairs of points and might be expected to be very similar. In practice, the ARI generally returns a number closer to 1 than the PNMI. The reason is that the ARI calculation ignores the off-diagonal terms, b (10) and c (01). In the PNMI calculation, the b term indicates the extent to which a pair sharing a lable in clustering A indicates that the pair will *not* share the lable in clustering B, and vice versa for the c term. Removing the b and c terms from the PNMI yields

$$\frac{\frac{a}{N}\log_2\frac{a}{N} - \frac{a}{N}\log_2[(\frac{a+b}{N})(\frac{a+c}{N})] + \frac{d}{N}\log_2\frac{d}{N} - \frac{d}{N}\log_2[(\frac{c+d}{N})(\frac{b+d}{N})]}{-\frac{a}{N}\log_2\frac{a}{N} - \frac{d}{N}\log_2\frac{d}{N}} \tag{3.7}$$

which has a similar form to the ARI:

$$\frac{\frac{a}{N} - (\frac{a+b}{N})(\frac{a+c}{N}) + \frac{d}{N} - (\frac{c+d}{N})(\frac{b+d}{N})}{\frac{m}{N} - (\frac{a+b}{N})(\frac{a+c}{N}) + \frac{N-m}{N} - (\frac{c+d}{N})(\frac{b+d}{N})} \tag{3.8}$$

By observation of these two equations, it is clear that both approach 1 as the $b \to 0$ and $c \to 0$, and approach 0 as $\frac{a}{N} \to (\frac{a+b}{N})(\frac{a+c}{N})$ and $\frac{d}{N} \to (\frac{c+d}{N})(\frac{b+d}{N})$. Figure 3.1.6 shows that the ARI, PNMI, and NMI all give similar estimates of cluster similarity, with ARI and PNMI being very similar and NMI being slightly pessimistic.

3.2 Multiple Membership Evaluation

The two intuitions we have for single membership clustering hold for multi-membership clustering, and can be restated as:

(a) If two multi-membership clusterings are similar, knowledge of the membership vector assigned to a point by C_1 should reduce the uncertainty about the membership vector assigned to the point by C_2.

(b) If two multi-membership clusterings are similar, knowledge that two points are assigned the same membership vectors by C_1 should reduce the uncertainty about the number of shared assignments given to the pair by C_2.

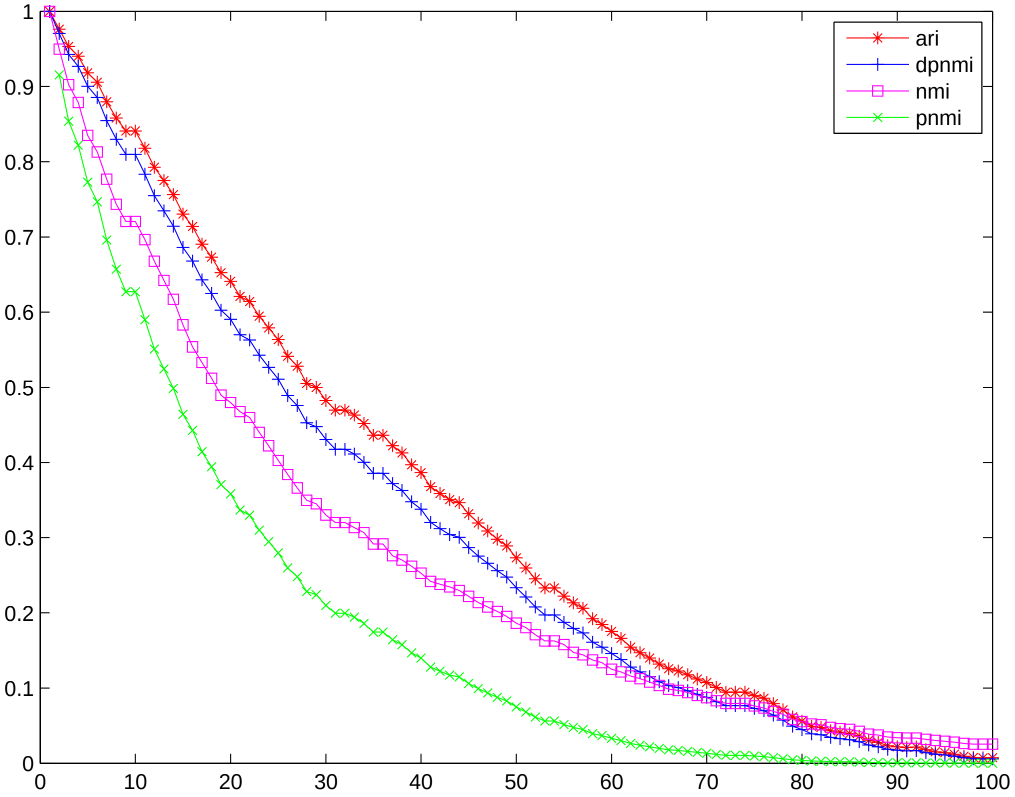

Figure 3.1: The empirical relationship between ARI, PNMI, and NMI for a dataset of 200 points and 10 clusters as the shared information between labellings changes. The vertical axis indicates the value of the comparison measure. The horizontal axis indicates the distortion between the two label matrices, which each tick indicating 0.5% of the labels changed.

3.2.1 A Probabilistic View of Multiple-membership Clustering

As with single-membership clustering, we can construct a probabilistic model of multiple-membership clustering. A multiple-membership clusterer assigns each data point a subset of labels from a set of k possible labels. This can be represented as a binary membership vector associated with each point where any number of the binary elements can be 1. Under the most naive assumption that assigned labels are independent of each other and of the features of the data point, the membership vectors can be represented as samples from a vector of Bernoulli random variables with means corresponding to the prior probability of cluster membership:

$$\begin{array}{ccccc} & C_1 & C_2 & \dots & C_k \\ \mathrm{M}=[& \mathrm{Bernoulli}(\rho_1), & \mathrm{Bernoulli}(\rho_2), & \dots & \mathrm{Bernoulli}(\rho_k) \quad] \end{array}$$

We would expect that a "good" multiple-membership clusterer would assign labels based as a function of the features:

$$\begin{array}{ccccc} & C_1 & C_2 & \dots & C_k \\ \mathrm{M}=[& f_1(\mathrm{features}(x)), & f_2(\mathrm{features}(x)), & \dots & f_k(\mathrm{features}(x)) \quad] \end{array}$$

In contrast to single-membership clustering, there is no constraint that $\sum_{i=1}^{k} f_i = 1$. Therefore, the presence of a particular label does not in general preclude the presence of any other label.

A multiple-membership clusterer assigns some subset λ of the set of labels, Λ ($|\Lambda| = k$) to each data point. Therefore, an input point can have any number of labels between 0 and k. As with single membership clustering, each point's labels can be represented by an indicator vector:

$$
\begin{array}{c ccccc}
\text{Label} & \multicolumn{5}{c}{\text{Indicator Vector}} \\
\text{label}(x_1) & 1 & 0 & \text{...} & 1 & 0 \\
\text{label}(x_2) & 0 & 0 & \text{...} & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\text{label}(x_{k-1}) & 1 & 1 & \text{...} & 1 & 0 \\
\text{label}(x_k) & 0 & 0 & \text{...} & 0 & 1
\end{array}
$$

The indicator vectors for a data set can be collected into a membership matrix M:

$$
\begin{array}{c c c}
\text{points} & \text{labels} & \text{M (array of indicator vectors)} \\
\begin{array}{c} x[1] \\ x[2] \\ x[3] \\ x[4] \\ \vdots \\ x[n-2] \\ x[n-1] \\ x[n] \end{array} &
\left[\begin{array}{c} L_2, L_5 \\ L_1, L_4, L_5 \\ L_1, L_5 \\ L_2 \\ \vdots \\ L_1, L_2 \\ L_2 \\ L_1, L_2, L_3 \end{array}\right] &
\left[\begin{array}{ccccc} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{array}\right]
\end{array}
$$

As with single-membership clustering, the priors for each cluster, ρ_i can be calculated as:

$$
\rho_i = p(\text{label}_i(x[m]) = 1) = \frac{\sum_{j=1}^{n} M[:, i]}{n}
$$

For multiple membership clustering, there is no requirement that $\sum_{i=1}^{k} M[j, :] = 1$; instead, $0 \leq \sum_{i=1}^{k} M[j, :] \leq k$. The edge cases, $\sum_{i=1}^{k} M[i, :] = 0$ and $\sum_{i=1}^{k} M[i, :] = k$ respectively represent the case of a point being assigned and the case of a point being assigned to all clusters.

For multiple membership clustering, there is no requirement that $\sum_{i=1}^{k} \rho_i = 1$; instead, $0 \leq \sum_{i=1}^{k} \rho_i \leq k$. Notably, the case

Of course, a "good" clustering would not be independent of the features, instead,

$$p(\text{label}_i(x[m]) = 1) = f_i(\text{features}(x[m])$$

with $0 \leq \sum_{i=1}^{k} f_i() \leq k$.

In contrast to single membership clustering, the lack of equality constraints on the sum of the ρ_i's and the f_i's allows these terms to be independent.

3.2.2 Mutual Information

In single-membership clustering, the mutual information measured how much knowing the labels assigned by one clusterer would improve one's knowledge of the labels assigned by a second clusterer. In multiple-membership clustering, each point is assigned a subset of labels. The direct analog to single-membership mutual information would be, then, to ask if knowing the subset of labels assigned to a point by one multiple-membership clusterer would provide information about the subset of labels assigned to the point by a second multiple-membership clusterer. This is equivalent to treating each point's binary membership vector as a unique label, and computing the mutual information over these labels.

Empirically, this approach does not work unless the number of data points, N, is much greater than the total number of unique labels, 2^k. For $N \leq 2^k$, the joint relationship between the labels cannot be established and the mutual information is artificially high.[2] This phenomenon is obvious in

---

[2] See Appendix for detailed explanation.

Figure 3.2.4, as even two completely random assignments of labels have an apparent mutual information of greater than 0.5.

An alternative is to rephrase the relevant question as “does the assignment of a particular label by clusterer A indicate the likely assignment of a particular label or group of labels by clusterer B?”. To answer this question, we consider each cluster independently, and compute a matrix of mutual informations:

	B_1	B_2	...	B_k
A_1	NMI($A_1; B_1$)	NMI($A_1; B_2$)	...	NMI($A_1; B_3$)
A_2	NMI($A_2; B_1$)	NMI($A_2; B_2$)	...	NMI($A_2; B_3$)
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
A_k	NMI($A_k; B_1$)	NMI($A_k; B_2$)	...	NMI($A_k; B_k$)

Each of these mutual information involves the calculation of the mutual information between two Bernoulli random variables which is a fast calculation that avoids the problem of too many labels. Depending on one’s expectation about the true number of clusters, the clusters in A and B can be aligned by trace maximization or bipartite graph partitioning (discussed in Chapter 4). Once the clusters are matched, the overall mutual information can be estimated as the trace of the NMI matrix divided by the number of clusters. As demonstrated in Figure 3.2.4, this quantity has much more reasonable characteristics than the mutual information computed over the indicator vectors.

3.2.3 Pairwise Mutual Information

The second intuition about comparing multiple-membership clusterings is that knowing the number of labels shared by two points under clustering A would be indicative of the number of labels shared by two points under clustering B. Note that this is somewhat more complicated than the single membership case. With multiple membership clustering, membership in one cluster in clustering A may be indicative of membership in more than one cluster in clustering B. The pairwise mutual information accounts for these possibilities.

As with single membership clustering, we first tabulate the the cooccurences of label counts between two clusterings. This tabulation is shown in Table 3.2. When normalized by the number of pairs, N, Table 3.2 represents the joint probability of shared label counts between clusterings A and B. The marginal distributions are the priors over label counts. The pairwise mutual information is then the KL divergence between the joint and the product of the marginals,

$$\begin{aligned} \mathrm{PMI}(A, B) &= \sum_{j=0}^{J} \sum_{k=0}^{K} p(j,k) \log_2\left(\frac{p(j,k)}{p(j)p(k)}\right) && (3.9) \\ \mathrm{PMI}(C_1, C_2) &= \sum_{j=0}^{J} \sum_{k=0}^{K} \frac{A_{j,k}}{N} \log_2\left(\frac{\frac{A_{j,k}}{N}}{\frac{(N_{j,.})(N_{.,k})}{N}}\right) && (3.10) \end{aligned}$$

3.2.4 The Multimembership Rand Index: The Omega Index

The Omega Index [CD88] extends the Adjusted Rand Index to overlapping clustering. In addition to counting the number of common pairs occurring together in 0 or 1 clusters, it also counts the number of pairs occurring together in 2,...,k clusters. Using the terms from Table 3.2,

	0	1	2	...k...	K	Marginal
0	A_0					$N_{0.}$
1		A_1				$N_{1.}$
2			A_2			$N_{2.}$
⋮						⋮
j				A_j		$N_{j.}$
⋮						⋮
J					A_J	$N_{J.}$
Marginal	$N_{.0}$	$N_{.1}$	$N_{.2}$	$N_{.k}$	$N_{.K}$	N

Table 3.2: Table for calculating multiple-membership pairwise mutual information and Omega Index

$$
\begin{aligned}
\Omega &= \frac{\sum_{j=0}^{min(J,K)} A_j}{N} \\
E[\Omega] &= \frac{\sum_{j=0}^{min(J,K)} N_{j.}N_{.j}}{N^2}
\end{aligned}
$$

As with the single-membership rand index, the omega index requires an adjustment to remove clusters sharing the same number of labels by chance.

$$
\begin{aligned}
\Omega_{adj} &= \frac{\frac{\sum_{j=0}^{min(J,K)} A_j}{N} - E[\Omega]}{1 - E[\Omega]} \\
&= \frac{observed\ index - expected\ index}{maximum\ index - expected\ index}
\end{aligned}
$$

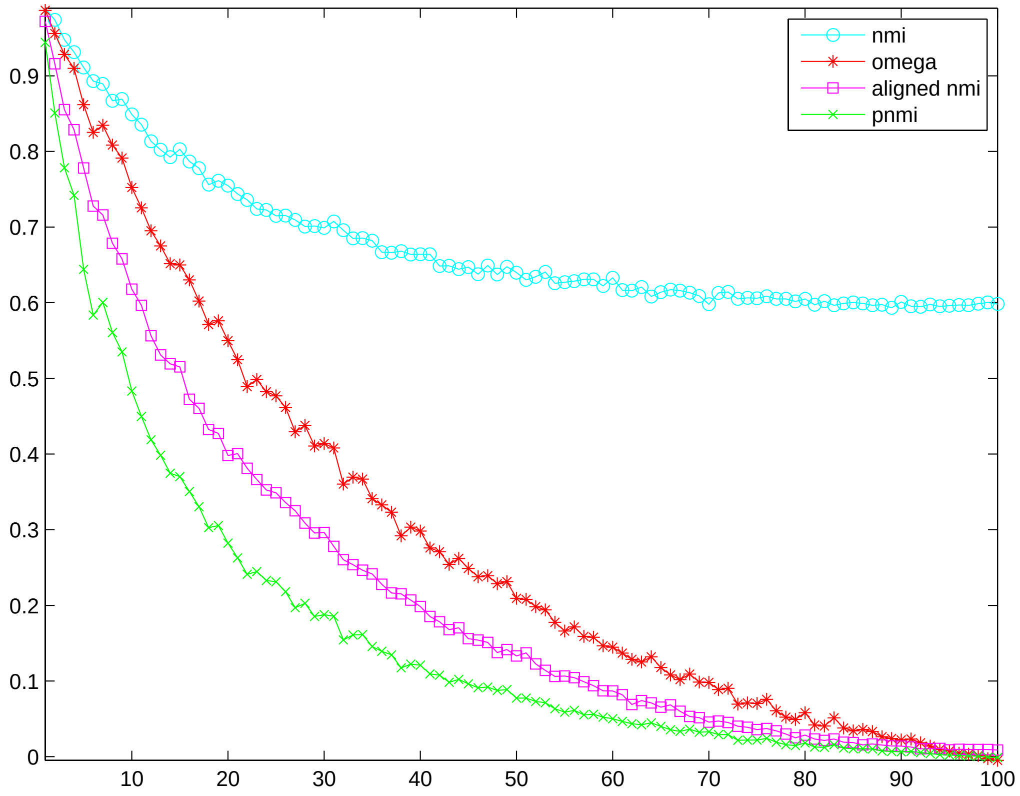

Figure 3.2: The empirical relationship between the Omega Index, PNMI, NMI, and aligned NMI for a dataset of 200 points labeled with 10 labels as the shared information between labellings changes. The vertical axis indicates the value of the comparison measure, and each step on the horizontal axis represents randomly assigning an additional 1% of the labels of the second clustering.

3.2.5 Omega Index and Multiple-membership PNMI

Figure 3.2.4 shows that the straightforward NMI is unusable for comparing multiple-membership clusterings. Of the other metrics, the Omega Index gives the most optimistic measure of multiple-membership similarity, while the PNMI and aligned NMI are very similar. Like the ARI in the single-membership case, the Omega Index considers only the diagonal terms - that a pair of points sharing m clusters in clustering A also shares exactly m clusters in clustering B - and ignores off-diagonal terms.

Chapter 4

Cluster Alignment

4.1 Introduction

Interpreting and comparing multiple-membership clusterings requires one to know the correspondence of labels between two clusterings or between a clustering and ground truth. Therefore, algorithms for *cluster alignment* are essential for the utility of multi-membership clustering. This chapter describes two ways of aligning multiple-membership clusterings, one using hypergeometric p-values and the other normalized mutual information, and introduces a visualization called the *cluster signature* that facilitates visual examination of cluster alignment results.

4.2 Aligning Nondisjoint Membership Matrices with Hypergeometric p-values

To align two binary membership matrices M_A and M_B, we seek a permutation of the columns of M_B that maximizes the similarity between each column of M_A and its corresponding column in M_B. A simple measure such as Hamming distance between the columns seems like a reasonable choice, but it implicitly assumes each column has the same density (ratio of 1's and 0's), which is not a valid assumption. To compensate for variations in density, we assume the 1's are independently distributed and calculate the probability of seeing the observed match given the densities of the binary vectors, v_1 (a column of M_A) and v_2 (a column of M_B):

With

- N as the vector length (total number of data points)
- d_1 as the number of 1's appearing in v_1, i.e. $d_1 = \sum_{i=1}^{N} v_1[i]$
- d_2 as the number of 1's appearing in v_2, i.e. $d_2 = \sum_{i=1}^{N} v_2[i]$
- S is the number of "overlapping" 1's, i.e. $S = \sum_{i=1}^{N} v_1[i] \times v_2[i]$
- the 1's in v_1 and v_2 uniformly distributed

the probability of seeing an observed overlap of S 1's is given by:

$$P(s = S) = \frac{\begin{pmatrix} N \\ d_1 \end{pmatrix} \begin{pmatrix} d_1 \\ S \end{pmatrix} \begin{pmatrix} N - d_1 \\ d_2 - S \end{pmatrix}}{\begin{pmatrix} N \\ d_1 \end{pmatrix} \begin{pmatrix} N \\ d_2 \end{pmatrix}} \tag{4.1}$$

where the denominator is the total number of permutations of the two vectors, and the numerator is the number of those permutations that have the observed overlap S:

- $\begin{pmatrix} N \\ d_1 \end{pmatrix}$ counts the number of ways d_1 1's can be placed in a vector of length N;
- $\begin{pmatrix} d_1 \\ S \end{pmatrix}$ counts the number of ways to choose the S overlapping points from the d_1 1's in v_1; and
- $\begin{pmatrix} N - d_1 \\ d_2 - S \end{pmatrix}$ counts the number of ways to place the remaining 1's in vector 1 such that they do not overlap with 1's in v_2.

While the denominator is obviously symmetric in assignment of v_1 and v_2, the numerator is not so clearly symmetric. Expansion of the binomial in Equation 4.1 coefficients makes it clear that the numerator is symmetric:

$$\frac{N!}{(d_1 - S)!S!(N - d_1 - d_2 + S)!(d_2 - S)!} \tag{4.2}$$

Algebraic simplification of Equation 4.1 yields:

$$P(s = S) = \frac{\binom{d_1}{S}\binom{N - d_1}{d_2 - S}}{\binom{N}{d_2}} \tag{4.3}$$

Equation 4.3 calculates the probability that two binary vectors v_1 and v_2 with densities d_1 and d_2 respectively will have S matched 1's (Equation 4.3 is the hypergeometric distribution evaluated at $s = S$). If we observe two N-length binary vectors, then

$$p - \text{value} = \sum_{s=S}^{s=\min\{d_1, d_2\}} P(s) \tag{4.4}$$

The p-value defined in Equation 4.4 gives the total probability of seeing the observed overlap (S) or a greater overlap. This value essentially measures the likelihood of the observed overlap being a random event, hence a small p-value indicates a small probability of seeing the observation at random. [1]

Returning to our original goal of aligning the columns of two membership matrices M_1 and M_2, it is clear that if we find that the p-value of matching

---

[1] A similar measure called the S-measure has been used previously in [LD84, LD85].

between column i of M_1 and column j of M_2 is very small, we can surmise that those two columns represent the same cluster. Finding the best possibly global matching of columns is an instance of the *Stable Marriage* problem [GI89], which is known to be NP-complete. The well-known Gale-Shapley algorithm [GS62] gives asymmetric solutions, so we opt to use a simple greedy matching algorithm.

1. Find the pairwise alignment p-value for every column of M_1 matched with every column of M_2. (For M_1 and M_2 each having k columns, this operation will take $(\frac{k^2}{2} - k)$ p-value calculations.)
2. Match the pair of columns $M_1[:, i]$ and $M_2[:, j]$ with the lowest pairwise p-value.
3. Repeat 1-2 until all columns have been assigned.

Note that this algorithm does not allow a single cluster in one clustering to be represented by a combination of clusters from the other clusterer.

4.3 Visualization of p-value based alignment

Along with the overlapping correspondence matching algorithm, we have developed a visualization tool which clearly presents correspondence alignments and other information. The visualization, as shown in Figure 1, presents three frames. The first two show cluster "signatures", and the last a bar chart of overlap p-values.

Cluster signatures are a visual representation of a nondisjoint label assignment designed to facilitate quick inspection and comparison of labelings.

Given an $n \times m$ membership matrix M, a cluster signature is constructed as follows:

1. For each row i,
   - (a) construct a vector t containing the indices of the 1's and set score$[i] = 0$
   - (b) If length$(t) = 0$, score$[i] = n + 1$
   - (c) if $t[2] - t[1] \neq 1$, score$[i] = t[1] + \sum_{k=2}^{\text{length}(t)} \frac{t[k]}{m^k}$
   - (d) if $t[2] - t[1] = 1$, score$[i] = t[1] + (1 - \frac{1}{m^{m+1}}) + \sum_{k=2}^{\text{length}(t)} \frac{t[k]}{m^{m+k}}$
2. Sort rows by increasing score

Ordering the membership matrix as described above puts the data points with no memberships at the bottom (step 1b) and sorts data points with memberships into blocks according to their minimum cluster label (steps 1c and 1d). Additionally, step 1d creates a visual overlap between consecutive overlapping clusters. Both steps 1c and 1d place points with additional cluster memberships into a unique order. This algorithm leads to a unique ordering for a given membership matrix, and is independent of the input order of the points.

4.4 Comparison of Clustering Methods

We present the alignment and visualization with a comparison of two algorithms, Model-based Overlapping Clustering (MOC) [BBK+05] and thresholded soft k-means [2] . MOC takes as input an observed $n \times m$ data matrix

---

[2] Soft k-means is the application of the expectation maximization (EM) algorithm to a mixture of k spherical Gaussians. Soft k-means minimizes an objective function equivalent

E and factors it into an $n \times k$ binary matrix M and an $m \times k$ (real) activation matrix A. Soft k-means is very similar to the standard k-means algorithm, except that points are given partial ("soft") assignment to centers. The resulting membership matrix is real, with the property that for any row j, $\sum_{i=1}^{m} x[i,j] = 1$. A soft clustering can be converted into a hard clustering by thresholding the soft membership matrix.

For ease of explanation and analysis, we demonstrate the application of our alignment and visualization on synthetic data. We generated a 10 cluster synthetic dataset using the MOC generative model, which is a conceptual representation of the biological and experimental processes that produce collections of microarray experiments. The MOC model assumes that an observed $n \times m$ data matrix E can be expressed as the product an $n \times k$ (binary) nondisjoint membership matrix M and an $m \times k$ (real) activation matrix A. For this example, we have used $n = 1000, m = 30, k = 10$. (Chapter 5 provides an explanation of the MOC model.)

Figure 1 illustrates the p-value based alignment of Model Based Overlapping Clustering with the ground truth for a synthetic data set. The uppermost box shows the signatures of each of the 10 clusters of the ground truth labels. The points have been sorted as described in Section III. The next box shows the signatures of each of the 10 clusters of the MOC labelling, with the points sorted as above and the clusters aligned. The final box shows the p-values of each of the alignments.

---

to the fuzzy c-means [Bez81] objective with "fuzziness" parameter m set to 1 and with dimensional scaling matrix A_k as identity. These are reasonable - but not necessarily optimal - parameters for this algorithm on our dataset. For a study on choosing m and A_k for a given dataset, see [DK03].

By visually comparing the first and second frames (the cluster signatures of the ground truth and of MOC's labelling, respectively) in Figure 1, one can observe that this clusterer, MOC, has found a cluster labelling that corresponds well to the actual clusters in the data. Columns 1,3,5, and 6 show log_{10} p-values of less than -110, indicating infinitesimal odds of those matches occurring by chance. The other columns show good alignment, although the imperfections in the result are evident. Overall, this alignment is very good, which is to be expected since the data was generated using the clusterer's generative model.

Figure 2 shows the alignment of a soft k-means clusterer run on the same artificial data set and thresholded at 0.3. Both the signature visualization and the p-value chart show that the clustering does not match the truth as well as the MOC clustering. While several clusters show reasonable correspondence with the ground truth, two clusters - columns 2 and 4 - completely fail to match.

4.5 Alignment for Cluster Ensembles

When the underlying generative model is unknown, combining the results of several diverse clusterers often improves the overall clustering result. One method of aligning clusters ensemble techniques is matching label assignments from each clusterer in the ensemble. Effective methods exist for disjoint membership matrices [SG02]; however, such methods are not applicable to nondisjoint membership matrices.

The p-value alignment method described in Section II provides a means of combining overlapping clusterings where each constituent clusterer uses the same k. In this section, we present results from combining three overlapping

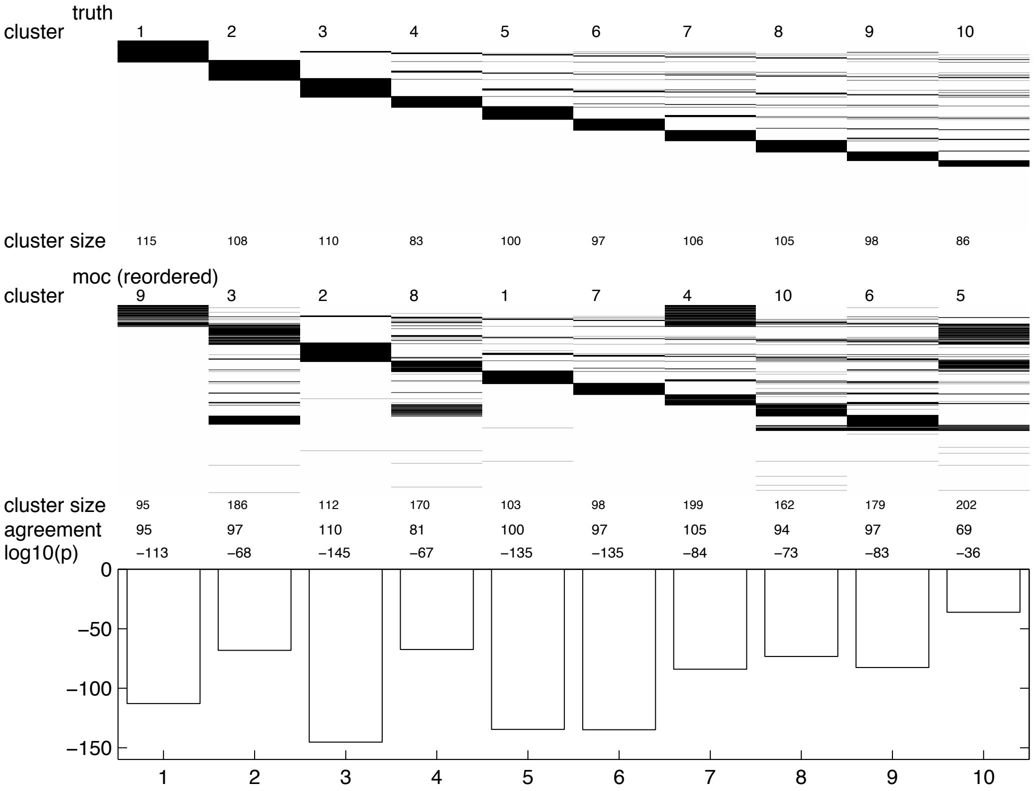

Figure 4.1: Visualization of correspondence between ground truth and MOC overlapping clustering for synthetic microarray data. The top frame is the "signature" of the ground truth, with the rows sorted as described in Section 4.3. The second frame is the "signature" of the MOC clustering, with the rows in the same order as in the top frame, and the columns matched using the algorithm described in Section 4.2. The final frame is a bar chart of the alignment p-values. The more negative the alignment p-values, the less likely the alignment happened due to random chance.

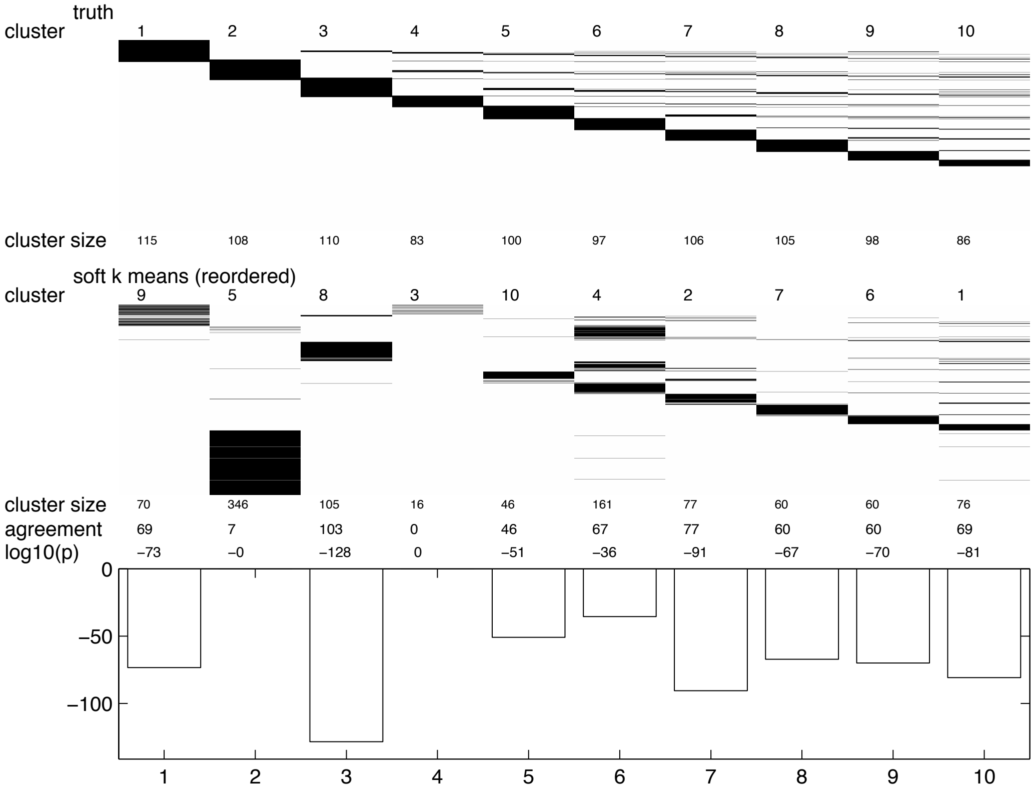

Figure 4.2: Visualization of correspondence between ground truth and Soft k-means overlapping clustering for synthetic microarray data.

clusterers - MOC, thresholded soft k-means, and gene-shaving - on the previously described synthetic microarray dataset.

The idea behind cluster ensembles is that each constituent clusterer will return a noisy representation of the actual underlying clustering. Combining several clusterers in an ensemble averages out the noise and often provides a better estimate of the underlying clustering. Clustering algorithms in general return clustering information in arbitrary order, necessitating cluster matching prior to ensemble operations.

We applied MOC, soft k-means thresholded at 0.3, and gene shaving to our synthetic microarray dataset. We aligned the results to each other, as shown in Figure 4.3. We then performed a majority-vote combination; that is, if a gene is marked as belonging to a cluster m by 2 out of the 3 clusters, we assign that gene to cluster m in the final result. Figure 4.4 shows the final consensus result aligned to the ground truth.

The consensus clustering shown in Figure 4.4 is superior to any of the constituent clusterings shown in Figure 4.3. It should also be noted that while the individual alignments for MOC and soft k-means shown in Figures 1 and 2 have some lower alignment p-values, the overall ensemble result appears to be much less noisy.

4.6 Conclusion

Overlapping clustering techniques provides a means of clustering microarray data in a way that matches nicely with biologic intuition about the participation of genes in biological processes. The results of overlapping clusterings, though, can be difficult to interpret. In this chapter, we presented a

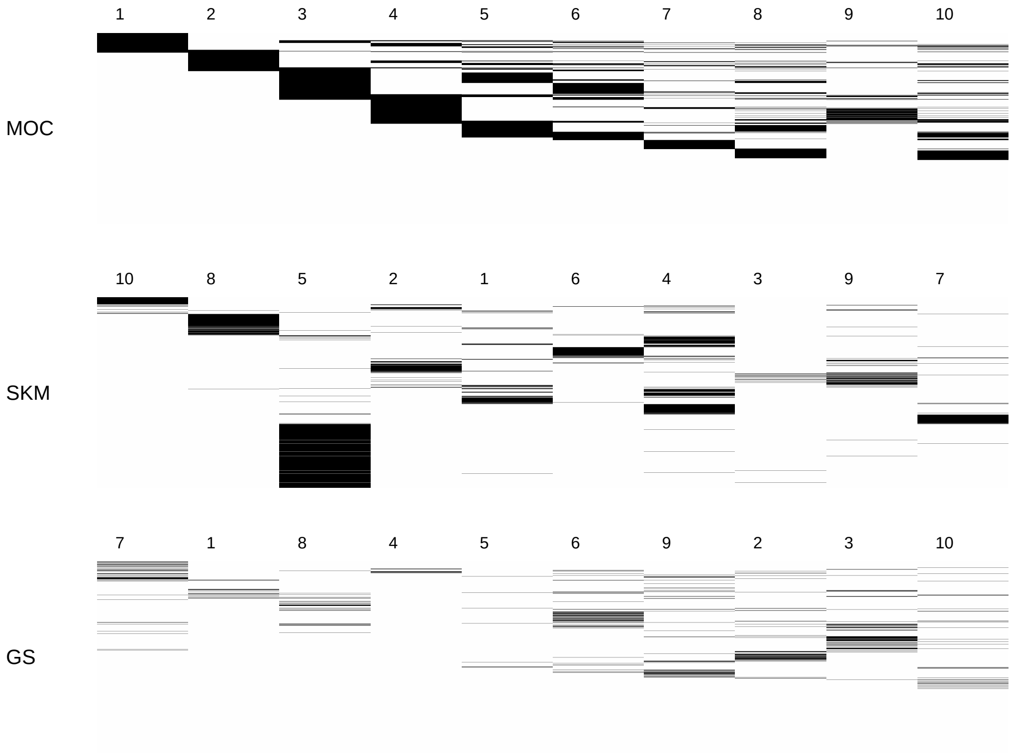

Figure 4.3: Visualization of aligned cluster signatures of MOC, soft k-means, and gene shaving clustering results on the synthetic data.

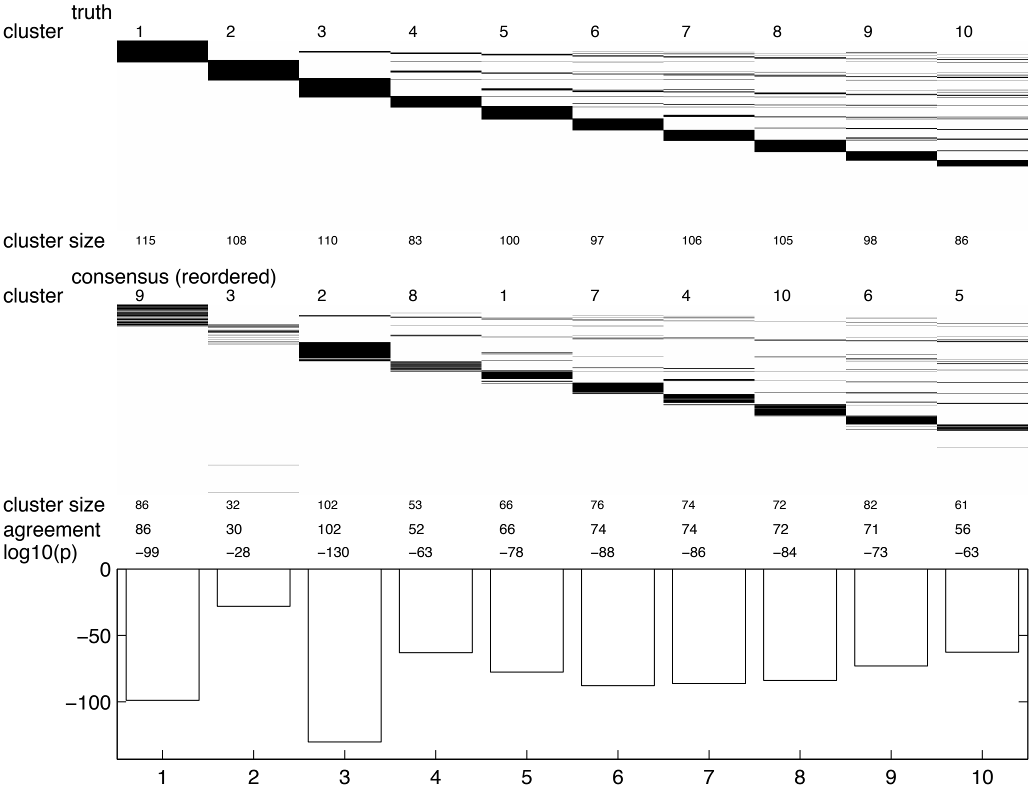

Figure 4.4: Visualization of ground truth cluster labels and aligned majority-vote consensus of MOC, soft k-means and gene shaving. The consensus recovers the actual labelling better than any of the individual clusterers.

cluster alignment method and a visualization tool which facilitates comparison, evaluation, and combination of overlapping clustering results.

Chapter 5

Model Based Overlapping Clustering

5.1 Introduction

As mentioned previously, the fundamental approaches to single membership clustering come in two flavors: model-based approaches which assume an underlying generative model and employ global minimization to find the model parameters, and non-model-based approaches which assume no model and find groupings based local information present in the data. This chapter describes a model-based approach to overlapping clustering; Chapter 6 describes a non-model based approach.

Gene expression data on microarrays has been a motivating domain for several data analysis techniques, including some of those aimed at overlapping clustering discussed in Chapter 2. A single microarray experiment captures the response (increased expression *vs* decreased expression) of many (∼10k) genes to some challenge to an organism's homeostasis; several individual experiments taken together allows the representation of each gene as a vector of its expressive responses to various stimuli. This arrangement allows the clustering genes into functional groups according to their expression profiles. While the standard single-membership approaches of k-means and hierarchical agglomerative have proved useful for initial analysis, it is known from wet lab experiments that many genes participate in several genetic pathways - a situation that can not be captured by single-membership clustering. A deeper understanding of the biology underlying the observed expression requires a model that both allows overlap and approximates the biological structure underlying cellular processes.

Such a model was introduced by Segal et al. [SBK03b]. This model, henceforth referred to as the SBK model, was presented as an instantiation of a Probabilistic Relational Model [FGKP99] intended specifically for clustering gene expression data. When extracted from the terminology of PRN's, the model is fundamentally an additive mixture model which represents a gene's observed expression level under a particular experimental condition as a sum of samples from unseen "processes" active in the experimental condition and involving the gene. This model, involving a binary membership matrix (*genes* × *processes*) and a real process activation matrix (*processes* × *conditions*), allows overlap by placing no constraint on the *genes* × *processes* binary membership matrix and in its structure provides a reasonable approximation of the underlying biology.

When viewed as an additive mixture model which allows overlap, the structural form of SBK model appears appropriate for several other clustering problems likely to involve overlapping memberships, notably in the domains of document clustering (where the microarray-specific entities of gene, process, and condition can be readily adapted to the appropriate document entities of word, topic, and document, respectively) and recommender systems (where gene, process, and condition become reviewer, item features, and item). These domains, however, are known to involve high-dimensional sparse data which is not well represented by the Gaussian models and Euclidean distances inherent in the SBK model as it was originally developed for microarray analysis. Adapting the model for use in these domains requires that it be generalized to work with any regular exponential family distribution and corresponding Bregman divergence. Further, this generalization creates the need for a general algorithm for monotonically improving the objective function of an overlapping model employing any regular exponential family distribution.

This chapter presents both the exponential family generalization of the SBK model, called "MOC", along with an algorithm for monotonically improving its objective function. The effectiveness of the generalized model and update algorithm are demonstrated on experiments involving subsets of the *20-Newsgroups* and *EachMovie* data sets. We compare the MOC model's results to an alternative "straw man" algorithm which produces a multiple membership clustering by first producing a standard probabilistic mixture model "soft" clustering and then making a hard assignment of each item to one or more clusters using a threshold on the cluster membership probability. While a very straightforward way to generate multiple memberships, the ability of thresholded soft clustering to produce quality overlapping clusterings is questionable, as the underlying mixture model assumes single membership. Our experiments show that the generalization of the SBK model, MOC, which is inherently a multiple-membership model, produces groupings that are more similar to the ground truth overlapping categories on the experimental datasets.

5.2 Models

In this section, we describe three models: the SBK model, the "straw-man" model which finds overlapping clusterings by thresholding a standard mixture model, and the MOC model which generalizes the SBK model.

5.2.1 The SBK Model

Probabilistic Relational Models (PRMs) [FGKP99] extend the basic concepts of Bayesian networks into a framework for representing and reasoning with probabilistic relationships between entities in a relational structure. The SBK model is an instantiation of a PRM for capturing the relationships between genes, processes, and measured expression values on DNA microarrays. The structure of the instantiated model succinctly captures the underlying biological understanding of the mechanism generating the observed microarray values — namely, that genes participate in processes, experimental conditions cause the invocation of processes at varying levels, and the observed expression value in any particular microarray spot is due to the combined contributions of several different processes. The SBK model places no constraints on the number of processes in which any gene might participate, and thus gene membership in multiple processes, i.e., overlapping clustering, naturally follows.

The SBK model employs three matrices: the observed real expression matrix X *(genes × experiments)*, a hidden binary membership matrix M *(genes × processes)* containing the membership of each gene in each process, and a hidden real activity matrix A *(processes × conditions)* containing the activity of each process for each experimental condition. The model assumes that the expression value $X[i,j]$ corresponding to gene i in experiment j has a Gaussian distribution with constant variance. The mean of the distribution is equal to the sum of the activity levels $A[h,j]$ of the processes h in which gene i participates so that

$$p(X[i,j]|M_i,A) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{1}{2\sigma^2}(X[i,j]-M[i,:]A[:,j])^2).$$

The SBK model further assumes that M and A are independent so that $p(M, A) = p(M)p(A)$ and that $X[i, j]$'s are conditionally independent given $M[i, :]$ and $A[:, j]$. M and A are assumed to be component-wise independent as well.

Assuming that elements of A are uniformly distributed, and noting that the conditional distribution of elements of X is Gaussian, the log-likelihood of the joint distribution over X, M, and A is

$$\max_{M,A} \log p(X, M, A) \equiv \min_{M,A} \left[\frac{1}{2\sigma^2} \|X - MA\|^2 - \log p(M) \right].$$

To find the value of the hidden variables M and A, the SBK model uses the iterative *Expectation Maximization (EM)* approach [DLR77]. The E step involves finding the best estimates of the binary genes-process memberships M. The M step involves computing the prior probability of gene membership in each process $p(M)$ and the process-condition activations A. The core parameter estimation problem is clearer when recast as a matrix decomposition problem (initially ignoring the priors). With the knowledge that there are k relevant processes in the observations, the goal is to find a decomposition of the observed expression matrix $X \in \mathbb{R}^{n \times d}$ into a binary membership matrix $M \in \{0, 1\}^{n \times k}$ and a real valued activation matrix $A \in \mathbb{R}^{k \times d}$ such that the reconstruction error $||X - MA||^2$ is minimized. Hence, the problem is one of matrix factorization not unlike SVD, but complicated by the constraint that M must be a binary matrix. Enforcement of this constraint on M requires special attention on each EM iteration. The procedure, which involves relaxing the binary constraint on M, thresholding M to binary and then tuning by local search proceeds as follows:

1. M is seeded with an initial estimate of the clustering in the data, usually

the output of a partitional clustering such as hierarchical or k-means run on the rows of X.

2. Next, the least-squares approximation of A for the given X and M is found as $A = M^{\dagger}X$, where $M^{\dagger}$ is the pseudo-inverse of M.
3. Using the A from step 2, the next approximation of M is found by relaxing the requirement that M be binary and solving a bounded least squares optimization for each gene in M. This effectively seeks a solution $\hat{M_i} = [0,1]^k$ for each row such that $||X_i - \hat{M_i}A||^2$ is minimized.
4. A binary solution M is then recovered from the real-valued solution $\hat{M}$ found in step 3 by thresholding. Since thresholding potentially moves the solution away from optimal, a local search is performed over every possible 0-flip of the post-threshold 1's to find the $M_i = \{0,1\}^k$ that minimizes $||X_i - M_iA||^2$.
5. Using the new M calculated in step 4, steps 2-4 are repeated until $||X - MA||^2$ is less than the desired convergence criteria.

Upon convergence of this procedure, the discovered binary gene-process memberships and real-valued process-condition activations will have settled into values that best reconstruct the observed microarray expression values in a least squares sense.

5.2.2 Overlapping Clustering by Thresholding a Mixture Model

Mixture models are commonly used for soft clustering, where each point is assigned a vector of probabilities corresponding to its probability of being generated by each generative component distribution. A hard partitional clustering can be then easily obtained by assigning each point to the component most likely to have generated it. The k-means algorithm is a specific case of a mixture model (using Gaussian components) with each point hard assigned to its most probably generating distribution.

In general, mixture models assume that given a set of n data points $\{X[i,:]\}_{i=1}^{n}$ in $\mathbb{R}^d$, represented by a $n \times d$ matrix X, fitting a mixture model to X is equivalent to assuming that each data point $X[i,:]$ is drawn independently from a probability density $p(X[i,:]|\Theta) = \sum_{h=1}^{k} \alpha_h p_h(X[i,:]|\theta_h)$, where $\Theta = \{\theta_h\}_{h=1}^{k}$, k is the number of mixture components, p_h is the probability density function of the h^{th} mixture component with parameters θ_h, and α_h are the component mixing coefficients such that $\alpha_h \geq 0$ and $\sum_{h=1}^{k} \alpha_h = 1$. In mixture model estimation, each point $X[i,:]$ is assumed to be generated from only one underlying mixture component. Let Z be a $n \times k$ boolean matrix such that $Z[i,j]$ is 1 if the j^{th} component density was selected to generate $X[i,:]$, and 0 otherwise. Let z_i be a hidden random variable corresponding to the index of the 1 in each row $Z[i,:]$; every z_i is therefore a multinomial random variable, since it can take one of k discrete values. Since the Z matrix is unknown, the optimum parameters Θ of the mixture model can be obtained using Expectation Maximization [DLR77]. The probability value $p(z_i = h|X[i,:],\Theta)$ after convergence of the EM algorithm gives the probability of the point $X[i,:]$ being generated from the h^{th} mixture component. Using these probabilities, mixture models are often used to generate a partitional clustering of the data, where the points estimated to be most probably generated from the h^{th} mixture model component are considered to constitute the h^{th} partition.

To produce an overlapping clustering from a mixture model where a row in Z can contain more than one 1, one can choose a threshold value λ such that $X[i,:]$ belongs to the h^{th} partition if $p(z_i = h|X[i,:],\Theta) > \lambda$. Such a thresholding technique can enable $X[i,:]$ to belong to multiple clusters; however, there are two notable problems with this method. The first is that the threshold parameter λ is difficult to deduce given only X. Secondly, this is not a natural generative model for overlapping clustering. In the mixture model, the underlying model assumption is that a point is generated from only one mixture component, and $p(z_i = h|X[i,:],\Theta)$ simply gives the probability of $X[i,:]$ being generated from the h^{th} mixture component. An overlapping clustering model should instead incorporate the idea that a point $X[i,:]$ could be generated by multiple mixture components simultaneously.

5.2.3 The MOC Model

The SBK model described above minimizes the squared loss (corresponding to an assumed Gaussian noise model) between X and MA, and is not amenable to estimating the optimal M and A corresponding to other loss functions. In MOC, we generalize the SBK model to work with a broad class of probability distributions, and provide an alternate minimization algorithm for the general model.

As stated previously, the SBK model and thus the MOC model have the characteristic form of mixture models, but with the important difference in the relaxation of the multinomial constraint on the matrix Z, allowing Z to be an arbitrary boolean matrix. In the MOC model, we denote this unconstrained boolean matrix as the membership matrix M to differentiate it from the multinomial-constrained matrix Z. Every point in the observed matrix $X[i,:]$ has a corresponding k-dimensional boolean membership vector $M[i,:]$

with the h^{th} component $M[i,h]$ of this membership vector being a Bernoulli random variable indicating whether $X[i,:]$ belongs to the h^{th} cluster. A row of the membership matrix M can encode any of the 2^k possible configurations of the cluster assignment of the corresponding point in X.

Let us now consider the probability of generating the observed data points in MOC. A is the activity matrix of this model, where $A[h,j]$ represents the activity of cluster h while generating the j^{th} feature of the data. The probability of generating all the data points is

$$p(X|\Theta) = p(X|M,A) = \prod_{i,j} p(X[i,j]|M[i,:],A[:,j]) \tag{5.1}$$

where $\Theta = \{M,A\}$ are the parameters of p, and $X[i,j]$'s are conditionally independent given $M[i,:]$ and $A[:,j]$. In MOC, we assume p to be the density function of any regular exponential family distribution, and also assume that the expectation parameter corresponding to $X[i,:]$ is of the form $M[i,:]A$, so that $E[X[i,:]] = M[i,:]A$. In other words we assume that each $X[i,:]$ is generated from an exponential family density whose mean $M[i,:]A$ is determined by taking the sum of the activity levels of the components that contribute to the generation of $X[i,:]$, i.e., $M[i,h]$ is 1 for the active components.

Using the above assumptions and the bijection between regular exponential distributions and regular Bregman divergences [BMDG04], the conditional density can be represented as:

$$p(X[i,j]|M[i,:],A[:,j]) \propto \exp\{-d_\phi(X[i,j],M[i,:]A[:,j])\} \tag{5.2}$$

where d_ϕ is the Bregman divergence corresponding to the chosen exponential density p. For example, if p is the Poisson density, d_ϕ is the I-divergence; if p is the Gaussian density, d_ϕ is the squared Euclidean distance [BMDG04].

Evaluating the model involves maximizing the joint distribution of X, M and A:

$$\begin{aligned} p(X, M, A) &= p(M, A)p(X|M, A) \;=\; p(M)p(A)p(X|M, A) \\ &= \left(\prod_{i,h} p(M[i, h])\right)\left(\prod_{h,j} p(A[h, j])\right)\left(\prod_{i,j} p(X[i, j]|M[i, :], A[:, j])\right) . \end{aligned}$$

Making similar model assumptions as in Section 5.2.1, we assume that M and A are independent of each other *a priori* and A is distributed uniformly over a sufficiently large compact set, implying that $p(M, A) = p(M)p(A) \propto p(M)$. Then, maximizing the log-likelihood of the joint distribution gives

$$\begin{aligned} \max_{M,A} \log p(X, M, A) &\equiv \max_{M,A} \left[\sum_{i,h} \log p(M[i, h]) - \sum_{i,j} d_\phi(X[i, j], M[i, :]A[:, j])\right] \\ &\equiv \min_{M,A} \left[\sum_{i,j} d_\phi(X[i, j], (M[i, :]A[:, j]) - \sum_{i,h} \log \alpha[i, h]\right] . \end{aligned}$$

where $\alpha[i, h] = p(M[i, h])$ is the Bernoulli prior probability of the i-th point having a membership $M[i, h]$ to the h-th cluster.

5.3 Algorithms and Analysis

In this section, we propose and analyze algorithms for estimating the overlapping clustering model given an observation matrix X. In particular, from a given observation matrix X, we want to estimate the prior matrix α, the membership matrix M and the activity matrix A so as to maximize $p(X, M, A)$, the joint distribution of (X, M, A). The key idea behind the estimation is an alternating minimization technique that alternates between updating α, M and A.

5.3.1 Updating α

The prior matrix α can be directly calculated from the current estimate of M. If π_h denotes the prior probability of any point belonging to cluster h, then, for a particular point i, we have $\alpha[i, h] = \pi_h^{M[i,h]}(1-\pi_h)^{1-M[i,h]}$. Since π_h is the probability of a Bernoulli random variable, and the Bernoulli distribution is a member of the exponential family, the maximum likelihood estimate is just the sample mean of the sufficient statistic [BMDG04]. Since the sufficient statistic for Bernoulli is just the indicator of the event, the maximum likelihood estimate of the prior π_h of cluster h is just $\pi_h = \frac{1}{n}\sum_i 1_{\{M[i,h]=1\}}$. Thus, one can compute the prior matrix α using these update equations.

5.3.2 Updating M

In the main alternating minimization technique, for a given X, A, the update for M has to minimize

$$\sum_{i,j} d_\phi(X[i, j], (M[i, :]A[:, j])).$$

Since M is a binary matrix, find its optimal assignment is an integer optimization problem which is not solvable by any known polynomial time algorithm. The explicit enumeration method involves evaluating all 2^k possibilities for every data point, which quickly becomes prohibitive for even moderate values of k. Finding an optimal M, then, requires an alternate approach.

The SBK model addressed this issue by the real relaxation of the problem allowing M to take real values in $[0, 1]$. For particular choices of the Bregman divergence specific algorithms can be devised that employ gradients in the real relaxed domain to find optimal solutions. For example, when the

Bregman divergence is the squared loss, the corresponding problem is just the bounded least squares (BLS) problem given by

$$\min_{M:0\leq M[i,h]\leq 1} \|X - MA\|^2,$$

for which there are well studied algorithms [Bjo96]. After convergence of the bounded least squares or other gradient optimization algorithm on the relaxed bounded real version of M, the binary membership constraints can be reinforced by rounding or thresholding [SBK03b]. Since rounding is in effect quantizing a point in real space onto a grid, the introduced quantization error can be a significant perturbation from the optimal point. The SBK model addresses this possibility by performing a local search around the quantized point by individually flipping each of the "on" clusters (1's in the membership matrix) "off" and evaluating the loss function. If a flipped solution produces a reduced loss, that solution is used instead of the original solution produced by rounding.

The SBK model performs this quantization with local search on each iteration of EM. An alternative would be to run EM to convergence and then perform quantization with local search, which would require a modification to the update equation for the priors, π_h and $\alpha[i, h]$. In either case, the real relaxation/quantization/local search approach requires that gradients be available for M and that a bounded least squares algorithm exist, which may not be the case for all loss functions.

A general approach to finding a binary M independent of loss function requires directly addressing the integer optimization problem without performing real relaxation. Such an approach needs to find a "good" solution (i.e. one that produces a low value of the loss function) by directly exploring the elements of the binary space of M. A brute-force, exhaustive search of a k-dimensional binary space would guarantee finding the optimal assignment, but would require 2^k evaluations, which is excessively computationally expensive for realistic values of k. As is often the case, a viable solution requires accepting some risk of a non-optimal solution for the benefit of vastly decreased computational expense.

To devise such a solution, we begin by making two observations regarding the problem of estimating M:

1. In most domains, we expect that a data point is more likely to be in very few clusters rather than most or all of them
2. For each data point i, estimating $M[i,:]$ is a variant of the *subset sum problem*

Regarding the first observation, for a domain where it is believed or desirable that each data point can belong to at most k_0 clusters, where $k_0 \leq \frac{k}{2}$ and $k_0 \ll k$, then it may be computationally feasible to perform an explicit search over all the possibilities, given the bound on the number of possible assignments:

$$\binom{k}{1} + \binom{k}{2} + \cdots + \binom{k}{k_0} \leq k_0 \left(\frac{ek}{k_0}\right)^{k_0} < 2^k.$$

However, in general, even this reduced brute-force search may only be feasible for very small values of k_0. Further, an apriori value of k_0 may not be available for a given problem, and an inappropriate choice could lead to a particularly poor solution for M.

Instead of relying on observation 1 to guide our strategy, it serves better as a heuristic to reduce the search space indicated by the second observation, that the assignment of M has the form of a subset sum problem. The subset sum problem is one of the hard knapsack problems [Chv80] intended to solve the following:

> Given a set of k natural numbers $a_1, \ldots, a_k$ and a target number x, find a subset S of the numbers such that $\sum_{a_h \in S} a_h = x$.

In practical scenarios, the values of objects available for combining into subsets are real numbers, and the goal becomes finding a subset such that the sum over the subset is the *closest* possible to x in terms of the appropriate loss function for the domain. In our case, closeness is measured using a Bregman divergence as the loss function and instead of one subset sum, our choice of M creates several sums ($M[i,:]A$) which must all simultaneously be close to their intended target values ($X[i,:]$). Stated explicitly, our task is to find $M^*[i,j]$ such that

$$\begin{aligned} M^*[i,:] &= \operatorname*{arg\,min}_{M[i,:] \in \{0,1\}^k} d_\phi(X[i,:], M[i,:]A) \\ &= \operatorname*{arg\,min}_{M[i,:] \in \{0,1\}^k} \sum_{j=1}^{m} d_\phi(X[i,j], \sum_{h=1}^{k} M[i,h]A[j,h]). \end{aligned}$$

Thus, there are m target values $X[i,1], \ldots, X[i,m]$, and for each target value $X[i,j]$ the subset is to be chosen from the subset objects with values $A[1,j], \ldots,$ $A[k,j]$. The total loss is the sum of the individual losses, and the problem is to find a single $M^*[i,:]$ that minimizes the total loss.

By employing the subset sum structure of the problem along with the heuristic suggested by observation 1, we devised a search algorithm to efficiently find good (but not necessarily optimal) assignments for M. The algorithm, *Knapsack Membership Search* (`KMS`), starts with one cluster turned “on” and greedily looks for the next best cluster to turn “on” so as to minimize the loss function. If such a cluster is found, then both clusters are set to “on” and the process repeats by holding those two clusters “on” and searching for a third whose activation reduces the loss. In general, if h clusters are turned “on”, `KMS` considers turning each one of the remaining $(k - h)$ clusters “on”, one at a time, and computes loss corresponding to the membership vector with $(h + 1)$ clusters turned “on”. If, at any stage, turning “on” each one of the remaining $(k - h)$ clusters increases the loss function, the search process is terminated. Otherwise, the algorithm picks the best $(h + 1)^{th}$ cluster to turn “on”, and repeats the search for the next best on the remaining $(k - h - 1)$ clusters.

This procedure of course depends on the order in which clusters are considered to be turned “on”. In particular, the choice of the first cluster to be turned “on” will partly determine which other clusters will get turned “on”. To resolve this permutation dependency, `KMS` should theoretically consider all $k!$ permutations of cluster ordering, compute the minimum loss in that permutation, and then return the minimum loss over all permutations. However, such an approach would make the algorithm unusable. Instead, we assume that searching a limited number of permutations provides sufficient coverage for finding an acceptable solution. `KMS` restarts its search k times, with each of the k clusters being the initial cluster once. Among all of the runs, as well as a trivial evaluation of the all-zero membership vector, the algorithm returns the cluster membership that minimized the loss function. With k runs, each with worst case $\mathrm{O}(k^2)$, the overall worst case running time is $\mathrm{O}(k^3)$.

5.3.3 Updating A

The real-valued activity matrix A has no restrictions on its values, making its update step much simpler than that for M. The only constraint that such an update needs to satisfy is that MA stays in the domain of ϕ. Each Bregman divergence has an associated update equation; those that are used in our experiments, the squared loss and the I-divergence, are provided in this section.

In the case of the square loss, since the domain of ϕ is $\mathbb{R}$, the problem

$$\min_A \ \|X - MA\|^2$$

is just the standard least squares problem that can be exactly solved by

$$A = M^{\dagger} X,$$

where $M^{\dagger}$ is the pseudo-inverse of M, and is equal to $(M^T M)^{-1} M^T$ in case $M^T M$ is invertible.

In the case of I-divergence or un-normalized relative entropy, the problem

$$\min_A d_I(X, MA) = \min_A \sum_{i,j} \left(X[i,j] \log \frac{X[i,j]}{(M[i,:]A[:,j]} - X[i,j] + M[i,:]A[:,j] \right) ,$$

has been studied as a non-negative matrix factorization technique [LS01]. The optimal update for A for given X, M multiplicative and is given by

$$A[h,j] = A[h,j] \frac{\sum_i M[h,i] X[j,i] / M[i,:]A[:,j]}{\sum_i M[h,i]}.$$

In order to prevent a division by 0, it makes sense to use $\max(M[i,:]A[:,j], \epsilon)$ and $\max(\sum_i M[h,i], \epsilon)$ as the denominators for some small constant $\epsilon > 0$.

With the above updates, the respective loss functions are provably non-increasing.

5.4 Experiments

This section describes the details of experiments demonstrating the performance of MOC *vs* the strawman thresholded mixture model on real-world data sets.

5.4.1 Methodology

Our evaluation experiments were performed on three different types of data: synthetic microarray-like data, movie recommendation data, and text documents.

Synthetic data: In [SBK03b], apart from demonstrating their approach on gene microarray data and evaluating on standard biology databases, Segal et al. also showed results on microarray-like synthetic data with a clear ground truth since the biology databases are generally believed to be lacking in coverage. Their synthetic data was generated by adding noise to points generated by the SBK model. Employing the same technique, we created three synthetic datasets of different sizes:

1. *small-synthetic*: a dataset with $n = 75$, $d = 30$ and $k = 10$;
2. *medium-synthetic*: a dataset with $n = 200$, $d = 50$ and $k = 30$;
3. *large-synthetic*: a dataset with $n = 1000$, $d = 150$ and $k = 30$.

For the synthetic datasets we used squared Euclidean distance as the cluster distortion measure in the overlapping clustering algorithm, since Gaussian densities were used to generate the noise-free datasets.

Movie Recommendation data: The EachMovie[1] dataset has 5-point user ratings for the 74,424 movies in the collection. The corresponding movie genre information is extracted from the Internet Movie Database (IMDB)[2] collection. If each genre is considered as a separate category or cluster, then this dataset also has naturally overlapping clusters since many movies are annotated in IMDB as belonging to multiple genres, e.g., Aliens belongs to 3 genre categories: action, horror and science fiction. We created 2 subsets from the EachMovie dataset: (1) *movie-taa*: 300 movies from the 3 genres – thriller, action and adventure; and (2) *movie-afc*: 300 movies from the 3 genres – animation, family, and comedy. We clustered the movies based on the user recommendations to rediscover genres, based on the belief that similarity in recommendation profiles of movies gives an indication about whether they are in related genres. For this domain we use I-divergence with Laplace smoothing as the cluster distortion measure.

Text data: Experiments were also run on 3 text datasets derived from the *20-Newsgroups* collection[3], which has 20,000 documents from 20 Usenet newsgroups. We processed the original newsgroup articles to recover the multiple newsgroup labels on each message posting. From the full dataset, a subset was created having 100 postings in each of the 20 newsgroups, from which the following three data subsets were created with varying levels of overlap in the topics: (1) *news-similar-3*; (2) *news-related-3*; and (3) *news-different-3*. Details of these datasets are outlined in [BBM04]. The vector-space model of each data subset was created using standard text pre-processing methods [DM01], and each data subset has 300 points in high-dimensional space ($>$ 1000 words).

---

[1] http://research.compaq.com/SRC/eachmovie

[2] http://www.imdb.com

[3] http://www.ai.mit.edu/people/jrennie/20Newsgroups

In this case, I-divergence was again used as the Bregman divergence for overlapping clustering, with suitable Laplace smoothing.

We used an experimental methodology similar to the one used to demonstrate the effectiveness of the SBK model [SBK03b]. For each dataset, we initialized the overlapping clustering by running k-means clustering, where the additive inverse of the corresponding Bregman divergence was used as the similarity measure and the number of clusters was set by the number of underlying categories in the dataset. The resulting clustering was used to initialize our overlapping clustering algorithm.

To evaluate the clustering results, precision, recall, and F-measure were calculated over pairs of points. For each pair of points that share at least one cluster in the overlapping clustering results, these measures try to estimate whether the prediction of this pair as being in the same cluster was correct with respect to the underlying true categories in the data. Precision is calculated as the fraction of pairs correctly put in the same cluster, recall is the fraction of actual pairs that were identified, and F-measure is the harmonic mean of precision and recall.

5.4.2 Results

Table 5.1 presents the results of MOC versus the standard mixture model for the datasets described in Section 5.4.1. Each reported result is an average over ten trials. For the synthetic data sets, we compared our approach to thresholded Gaussian mixture models; for the text and movie data sets, the baselines were thresholded multinomial mixture models. Table 5.1 shows that for all domains, even though the thresholded mixture model has slightly better precision in most cases, it has significantly worse recall: therefore MOC

	F-measure		Precision		Recall	
Data	MOC	Mixture	MOC	Mixture	MOC	Mixture
small-synthetic	**0.64** ± 0.12	0.36 ± 0.08	**0.83** ± 0.07	0.80 ± 0.07	**0.53** ± 0.14	0.24 ± 0.07
medium-synthetic	**0.71** ± 0.06	0.24 ± 0.01	**0.73** ± 0.05	0.60 ± 0.03	**0.70** ± 0.09	0.15 ± 0.01
large-synthetic	**0.87** ± 0.04	0.33 ± 0.01	0.85 ± 0.06	**0.87** ± 0.04	**0.89** ± 0.05	0.20 ± 0.01
movie-taa	**0.62** ± 0.03	0.50 ± 0.04	0.55 ± 0.01	**0.56** ± 0.01	**0.71** ± 0.07	0.46 ± 0.08
movie-afc	**0.76** ± 0.03	0.61 ± 0.07	0.80 ± 0.01	**0.81** ± 0.02	**0.72** ± 0.06	0.50 ± 0.09
news-different-3	**0.45** ± 0.01	0.41 ± 0.05	0.34 ± 0.01	**0.40** ± 0.05	**0.68** ± 0.05	0.41 ± 0.06
news-related-3	**0.54** ± 0.02	0.39 ± 0.02	0.42 ± 0.01	**0.44** ± 0.02	**0.76** ± 0.08	0.35 ± 0.01
news-similar-3	**0.35** ± 0.02	0.28 ± 0.01	0.23 ± 0.01	**0.24** ± 0.01	**0.69** ± 0.06	0.34 ± 0.01

Table 5.1: Comparison of results of MOC and thresholded mixture models on all datasets

	F-measure		Precision		Recall	
Data	KMS	BLS/search	KMS	BLS/search	KMS	BLS/search
small-synthetic	**0.64** ± 0.12	0.55 ± 0.20	0.83 ± 0.07	**0.98** ± 0.03	**0.52** ± 0.14	0.41 ± 0.19
medium-synthetic	**0.71** ± 0.06	0.65 ± 0.05	0.73 ± 0.05	**0.91** ± 0.06	**0.70** ± 0.09	0.51 ± 0.06
large-synthetic	**0.87** ± 0.04	**0.87** ± 0.02	0.85 ± 0.06	**0.92** ± 0.02	**0.89** ± 0.05	0.83 ± 0.04

Table 5.2: Results: `KMS` vs Bounded Least Squares (with search) for synthetic data

consistently outperforms the thresholded mixture model in terms of overall F-measure, by a large margin in most cases. Table 5.1 also shows that the performance of MOC improves empirically as the ratio of the data set size to the number of processes increases.

Table 5.2 compares the performance of using the `KMS` algorithm versus the bounded least squares (BLS) algorithm followed by local search, in the M estimation step in MOC. BLS/search gets better results on precision, which is expected since BLS is the optimal solution for the real relaxation of the M estimation problem for the Gaussian model. However `KMS` outperforms BLS/search on the overall F-measure. Moreover, BLS is only applicable for squared Euclidean distance, whereas `KMS` can be applied for M estimation with any distance function.

Detailed inspection of the results revealed that MOC finds overlapping clusterings that are closer to the ground truths for the text and the movie data. For example, for *movie-afc*, the average number of clusters a movie is assigned to is 1.19, whereas MOC clustering has an average of 1.13 clusters per movie. The thresholded mixture model got posterior probability values very close to 0 or 1, as is very common in mixture model estimation for high-dimensional data: as a result there was almost no cluster overlap for various choices of the threshold value, and points were assigned to, on average, exactly one cluster in the thresholded mixture models. MOC was also able to recover the correct underlying multiple genres in many cases, e.g., the movie "Toy Story" in the *movie-afc* dataset belongs to all the three genres of animation, family and comedy in this dataset, and MOC correctly put it in all 3 clusters.

Chapter 6

Similarity Space Overlapping Clustering

6.1 Introduction

When finding clusters in data, the fundamental information employed is the *similarity* between data elements. Similarity is an application-dependent function that takes two data elements and returns a number which indicates the degree to which the two elements are close with regard to the defined similarity measure. For mathematical convenience, the number returned by the similarity function is between 0 and 1, with 0 indicating the minimum similarity and 1 the maximum. Since a similarity exists for every pair of points, the set of all pairwise similarities is usually arranged into a square symmetric $n \times n$ matrix which is termed the *similarity space* of the data.

When devising a clustering algorithm it is critical to first define what is meant by a cluster. With a firm definition, the task of the algorithm is then to extract groups of data points which meet the definition. In a similarity space, a reasonable definition of a cluster is a group of points which are mutually more similar to each other than to points not in the group. Note that this definition allows an individual point or set of points within such a group to share high similarity with points in other groups and therefore equally merit group membership with them, thus exhibiting multiple-membership.

In this chapter, we first describe a probabilistic generative model of similarity spaces containing overlapping clusters. We then develop a theoretical multiple-membership implementation of complete-link hierarchical clustering, and offer a practical implementation. Next, we develop a hill-climbing iterative update technique for finding overlapping densities in similarity spaces, and demonstrate the performance of this algorithms on datasets of the structure described by the generative model.

6.2 Generative Model of Similarity Space Overlapping Clusters

6.2.1 Model Considerations and Background

When approaching the problem of finding overlapping clusters in data in a similarity space, it is important to first consider the nature of the process that generated the data. Toward this end, we examine the statistics of the similarity space, which are easily visualized via a histogram of the similarities between points. Histograms of similarity spaces are generally shaped like the histogram in Figure 6.1, which shows the similarity space of the Iris dataset. It is notable that most of the mass is close to zero - which is expected given that most points are not highly similar to most other points.[1] For a dataset such as Iris where the labels are known, we can further examine the distribution of similarities by producing histograms of the similarities of points sharing the same label and the similarities of points not sharing the same label. These distributions are shown in Figure 6.2. Notably, the distribution of similarities of points sharing a label has most of its mass distant from zero, while the distribution of points not sharing a label has most of its mass close to zero.

---

[1] A similarity space where most points display high similarity to most other points would contain very little information relevant for clustering, and would likely have been generated by a similarity function inappropriate for the data.

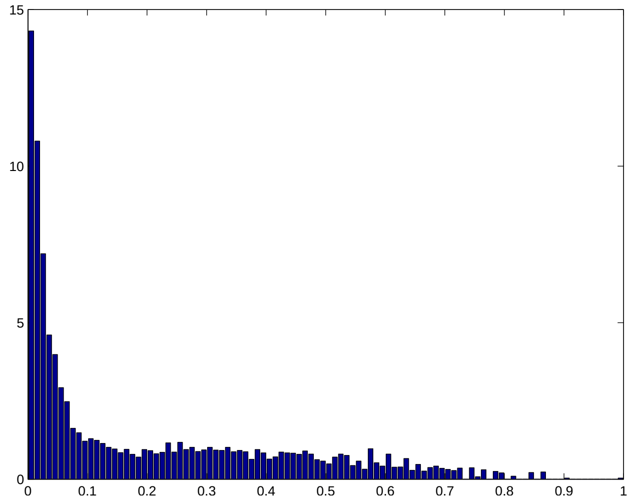

Figure 6.1: Histogram of the values in the similarity space of the iris dataset.

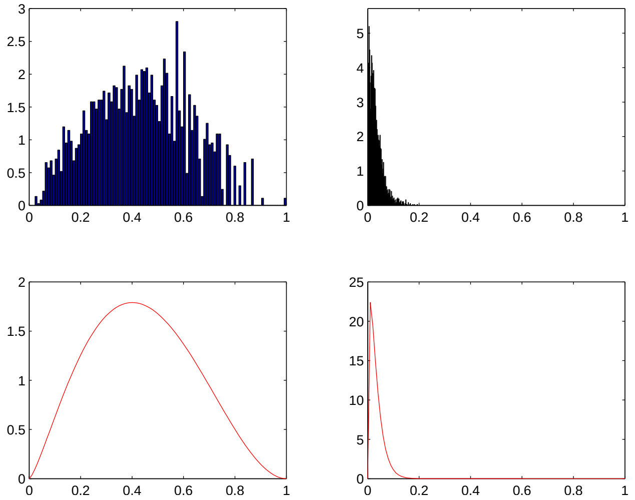

Figure 6.2: The upper row shows histograms of the within-cluster similarity and between-cluster similarity of points in the Iris dataset. Note that distribution of similarities of points not sharing a cluster has most of its mass near zero, while the distribution of similarities of points sharing a cluster has a center of mass far from zero. These two distributions can be approximated by beta distributions, shown in the second row.

These observations on the distributions of similarities in the Iris dataset provide intuition for the general structure of similarity spaces containing non-random data with clusters. Specifically, that elements of a similarity space can be viewed as being generated from a mixture of two beta functions such as those shown in Figure 6.4, one with most of its mass close to zero representing the "background" similarity between two points not sharing a cluster, and the other with most of its mass closer to one representing the high similarities of points sharing a cluster. This structure is in accordance with our general intuition about clusters in data - namely, that points sharing a cluster have higher similarity than those not sharing a cluster.

Now we consider how this property extends to multiple-membership clusters. The histogram of similarities illustrates the relationship of in-cluster similarities vs. between cluster similarities, but does not communicate anything about the membership structure of the clusters. The heatmap is the appropriate tool for visualizing membership structure. Similarity spaces in general, and graphs in general, can be visually displayed as heatmaps where the color of each point represents the weight of its corresponding arc. Figure 6.3 shows the heatmap for the Iris dataset. It is notable that in a single-membership clustering such as Iris, the true labels yield a heatmap which can be ordered such that the clusters form blocks along the diagonal, as are visible in the Iris heatmap. In a similarity space consisting of convex, non-overlapping clusters, the corresponding heatmap would consist of diagonal blocks of similarities drawn from the "within-cluster" distribution of similarities, and points outside of the diagonal blocks drawn from the "background" distribution. The problem of finding non-overlapping clusters can be posed as finding the matrix reordering that maximizes this property. For data containing multiple-membership clusters, such an ordering does not exist; for any block-diagonal reordering, there will continue to be off-diagonal "within-cluster" similarities corresponding to shared points' connections to clusters other than their neighbors in the reordering. The degree to which off-diagonal "within-cluster" points exist is indicative of the average overlap existing in the dataset.

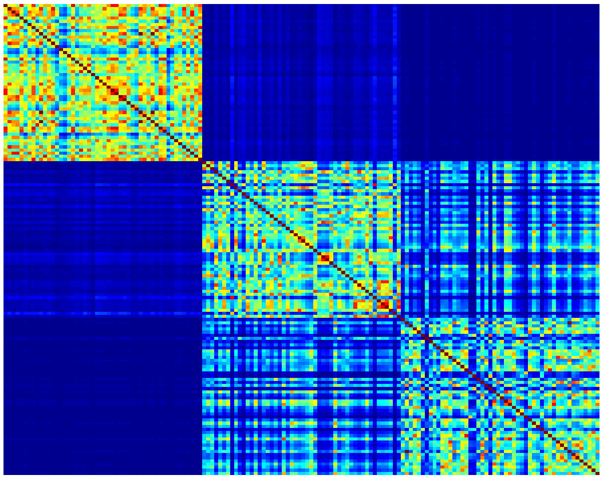

Figure 6.3: Heatmap of the similarity space of Iris. Notably, the similarity space is a symmetric matrix with ones along the diagonal and a notable block structure. Each block along the diagonal is a region of high similarity between points, i.e., a cluster. The middle and lower right cluster correspond to *Iris versicolor* and *Iris virginica* which are not linearly separable groups. The off-axis similarity indicates high similarity between points in the two groups.

A final component, which is not addressed by analyzing either the histogram or the heatmap of a similarity space, is the issue of single-link structure versus complete-link structure of the clusters. Complete-link clusters are convex[2]. Single-link clusters can be arbitrarily shaped, and are defined such that

---

[2] Convex here means that for a set A with minimum similarity (s-level) $s(A)$, i.e. $\forall i, j \in A, S[i,j] > s(A)$, A is convex if $\forall k \notin A, \exists i \in A$ such that $S[i,k] < s(A)$

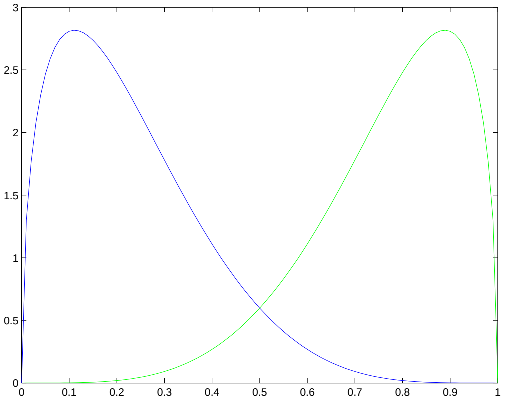

Figure 6.4: Values in the similarity space of a dataset containing clusters can be modeled as values drawn from one of two beta distributions. The distribution shown in blue with most of its probability mass close to zero represents the distribution of similarities of points not in the same cluster. We term this distribution the "background" and parameterize it as $\Theta_{background}$. The distribution shown in green with most of its probability mass close to one represents the distribution of similarities of highly similar points likely to be in the same cluster. We term this distribution the "cluster" distribution and parameterize it as $\Theta_{cluster}$.

within a cluster, two points may have a similarity of zero as long as they are linked by a path of in-cluster points where the similarity between any two points along the path is greater than the threshold s_{min}. When, as with complete-link, all of the clusters in a space are convex, all intracluster similarities come from the "within-cluster" similarity distribution. When clusters are not convex, some intracluster similarities may be drawn from the "background" distribution.

6.2.2 The Similarity Space Multiple-Membership Generative Model

Given the previously described properties of a similarity space containing clusters, we can design a generative model which produces a similarity

Parameter	Meaning
n	the number of points in the dataset
k	the number of clusters in the data
Q	a distribution over cluster sizes in $[1, n]$
c	a value in $[0, 1]$ indicating the complete-link character of the clusters
v	a value in $[0, 1]$ indicating the average overlap between clusters
$\Theta_{background}$	the parameters of the background Beta distribution
$\Theta_{cluster}$	the parameters of the within-cluster Beta distribution

Table 6.1: Parameters of the similarity space clustering generative model.

space containing overlapping clusters exhibiting those properties.[3] The generative model requires the parameters defined in Table 6.1. With those parameters specified, a similarity space containing n points and k clusters can be constructed as follows:

1. Initialize an $n \times n$ similarity matrix s such that the diagonal values are 1, and values in the upper triangular region are drawn from Beta($\Theta_{background}$).
2. For each of the k clusters,
    (a) Select a cluster size d by sampling from the distribution Q.
    (b) Construct the set A by drawing d points such that $\lceil d * v \rceil$ are randomly drawn from points already assigned to a cluster and $\lfloor d * (1 - v) \rfloor$ are randomly drawn from points not assigned to a cluster.
    (c) For every pair of points (i,j;$i < j$) in A, replace the point $s(i, j)$ with a sample from Beta($\Theta_{cluster}$) distribution with probability c.

---

[3] Note that in generating a similarity space representation of a dataset, we only require that similarities be symmetric. No assumptions are made about the original input space of the data.

Parameter	Value
n	500
k	10
Q	discrete Uniform over $(\lfloor \frac{n}{10} \rfloor, \lfloor \frac{n}{5} \rfloor)$
c	0.8
v	0.2
$\Theta_{background}$	(1.5,5)
$\Theta_{cluster}$	(10,3)

Table 6.2: Parameters of the example synthetic data set.

3. Reflect the upper triangular values of s over the diagonal so that s is symmetric.

We now examine an example similarity space generated according to the above algorithm using the parameter given in Table 6.1 corresponding to the Beta functions shown in Figure 6.5. This dataset's histogram and heatmap are shows in Figures 6.6 and 6.7 respectively. The histogram is notable for the smearing of the within-cluster similarity distribution toward lower similarity values. This smearing is due to the parameter c, which allows clusters to assume some single-link character. The heatmap, which is ordered according to a hierarchical agglomerative clustering, demonstrates that data with inherent overlapping clusters can not be ordered into a block-diagonal matrix.

6.3 Algorithms for Finding Clusters in Similarity Spaces

Given the data model described in the previous section, we now explore algorithmic approaches which in effect reverse engineer the generative model and thereby extract clusters from similarity space data. We first describe an extension of complete-link hierarchical clustering into the multiple-

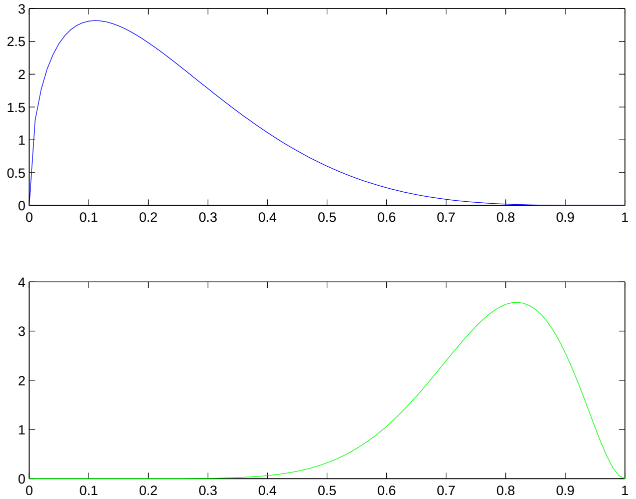

Figure 6.5: The "background" and "cluster" Beta functions with the parameters given in Table 6.2 used to generate the synthetic dataset.

membership framework and provide an exact but inefficient algorithm, followed by an adjusted algorithm that is feasible. Second, we describe a multiple-membership cluster finding algorithms that performs density searching in the similarity space.

6.4 Generalizing Complete-Link Hierarchical Clustering to Multiple Memberships

Hierarchical agglomerative clustering is conceptually the simplest method of clustering data. For data in a similarity space, the algorithm starts by labeling each point as a singleton cluster, and then proceeds to iteratively merge the most similar clusters until a single cluster encompasses all of the data. The bottom-up sequence of merges yields a hierarchical nesting of potential clusters which can later be selected as representatives of the data according to some criteria (e.g. number of clusters, stability of clusters, minimum similarity,

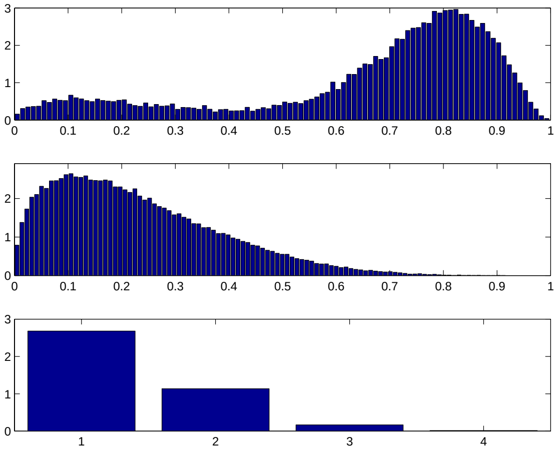

Figure 6.6: Characteristics of the synthetic data set generated using the parameters in Table 6.2. The first pane shows the distribution of within-cluster similarities, the second shows the background similarity distribution, and the bottom pane the distribution of memberships. Note that the within-cluster distribution is a mixture of the specified within-cluster Beta distribution and a small amount of the background Beta distribution. This is due to the parameter c which specifies complete-link character being less than one: points within a cluster are not required to have high similarity to all other points within the cluster.

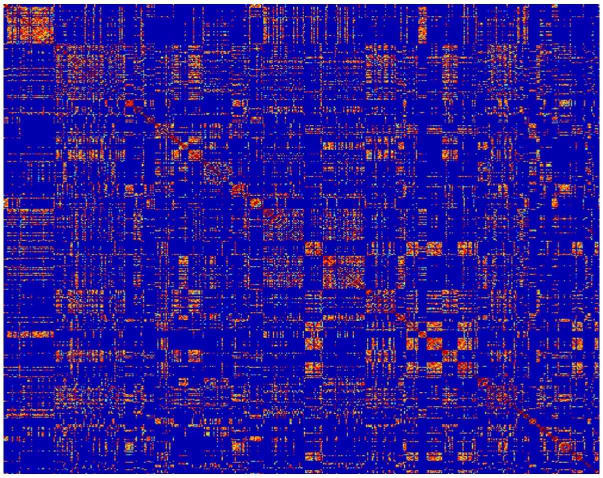

Figure 6.7: Heatmap of the synthetic data set generated using the parameters in Table 6.2. Due to the overlapping nature of the similarity space, the matrix can not be reordered so that the high similarities form a block-diagonal matrix.

etc.). By choosing a single merge pair at each iteration, hierarchical agglomerative clustering ensures that each data point belongs to only one cluster at a time. In domains where data is fundamentally overlapping, this property leads to clusters which do not match observations. Shepard and Arabie's seminal algorithm ADCLUS [SA79] was designed to overcome this shortcoming. ADCLUS, like hierarchical agglomerative, first generates a set of potential clusters which are then refined into data-representative clusters according to a given criterion.

The potential cluster generating step of the ADCLUS algorithm was the enumeration of *elevated subsets*, which are effectively the multiple-membership equivalent of the hierarchical agglomerative algorithm's single-membership nested clusters. Finding these elevated subsets requires a combinatorial search of the data involving significantly more computation than hierarchical agglomerative. Shepard and Arabie suggest an algorithm and provide references to others (e.g. [Con66]) which all share hierarchical agglomerative's bottom-up flavor. All are also questionable in their scalability; but the datasets at the time generally did not exceed 30 points, so the methods were quite serviceable.

Modern datasets are much larger, and the existing bottom-up procedures for enumerating elevated subsets are not feasible. In this section, we first describe a conceptual top-down algorithm for finding elevated subsets which is guaranteed to find all elevated subsets, but is infeasible due to combinatoric explosion. We then present a practical refinement of that conceptual algorithm.

6.4.1 Definition of *s*-level, ***elevated subset***, and ***rise***

A subset of data points can be characterized by the "spread" of the subset which is lower bounded by the similarity between the two least similar points in the set. The s-level of a subset of points is the minimum similarity among all pairs of points in the subset,

$$s(A) = \min_{i,j \in A}(S[i,j])$$

where $S[i,j]$ is the similarity between points i and j in the subset A.

A low s-level indicates that points in the subset are not very similar, while a high s-level indicates a cluster of highly similar points.

An *elevated subset* is a subset of points whose s-level is greater than that of any of its supersets, i.e., A is an elevated subset if

$$s(B) < s(A) \quad \forall B \supset A. \tag{6.1}$$

The *rise* of an elevated subset is the extent to which the s-level of subset A rises above the s-level of any of A's supersets:

$$r(A) = \min_{B \supset A}[s(A) - s(B)]$$

6.4.2 Generalizing complete-link hierarchical clustering

Several variations of the basic hierarchical clustering algorithm exist, each differing in the method of measuring similarity between clusters. The "complete-link" form measures similarity between clusters as the similarity between the least similar constituent points. This form is widely used due to its ability to produce convex clusters and to avoid other undesirable problems inherent in the other variations. Merges in the complete-link hierarchical agglomerative algorithm occur between the two clusters whose least similar points are more similar than any other pair of clusters. That is, clusters A_i and A_j are merged if

$$s(A_i \cup A_k) < s(A_i \cup A_j) \quad \forall k \neq j, \tag{6.2}$$

and

$$s(A_k \cup A_j) < s(A_i \cup A_j) \quad \forall k \neq i. \tag{6.3}$$

This merging procedure leads to a specific definition of what constitutes a cluster. From (6.2) and (6.3), it follows that

$$s(A_i \cup A_j \cup A_k) = \min[s(A_i \cup A_k), s(A_j \cup A_k)] < s(A_i \cup A_j) \quad \forall A_k \neq A_i, A_j. \tag{6.4}$$

If we rename the set $(A_i \cup A_j)$ A and the set $(A_i \cup A_j \cup A_k)$ B, then $B \supset A$ and from (6.4),

1.0000	0.1792	0.8256	0.9854	0.6611
0.1792	1.0000	0.8090	0.8935	0.5073
0.8256	0.8090	1.0000	0.5789	0.5729
0.9854	0.8935	0.5789	1.0000	0.4596
0.6611	0.5073	0.5729	0.4596	1.0000

Table 6.3: 5 points similarity space used to create the multiple membership dendrogram shown in Figure 6.8.

$$s(B) < s(A) \quad \forall B \supset A. \tag{6.5}$$

which defines A as an elevated subset from (6.1). Therefore, each merge in complete-link hierarchical clustering results in a cluster which is an elevated subset.

The bottom-up merging procedure of complete-link hierarchical clustering yields a nested series of elevated subsets such that a given point takes a single route through the hierarchy, belonging at most to one level-equivalent elevated subset at a time. Each merge is performed greedily, using the clusters resulting from previous greedy merges. Due to this local greedy aspect of the algorithm, elevated subsets as defined above occur outside of the hierarchical clustering tree, as shown in Figure 6.8.

6.5 Existing Algorithms for Finding Elevated Subsets

The existing methods of elevated subset enumeration [Con66, SA79] are all bottom-up algorithms that follow a similar procedure:

1. Sort the pairwise similarities in decreasing order.

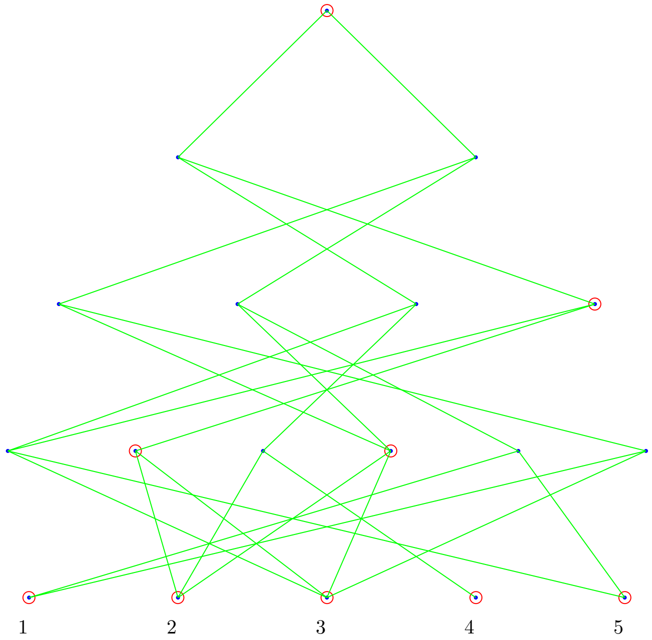

Figure 6.8: Dendrogram showing all elevated subsets and inheritance paths for the 5 point similarity space given in Table 6.3. Vertices corresponding to elevated subsets found by complete-link hierarchical clustering are circled in red. It is notable that the complete set of elevated subsets is a superset of the set of complete-link hierarchical clusters, and that the all-points cluster at the top and the singleton clusters at the bottom are shared between the algorithms.

2. Starting at the top with the largest pairwise similarity, threshold the similarities with that value.

3. On the resulting graph, find all cliques.

4. Proceed to the next pairwise similarity and repeat 2-4.

From examples in the literature [SA79], it appears that this algorithm is effective for datasets containing less than 30 points. However, the clique-enumeration step is an NP-hard problem, and thus does not scale well.

6.5.1 Exact Algorithm for Enumerating Elevated Subsets

The existing elevated subset enumeration algorithms directly follow hierarchical agglomerative and agglomerate subsets from the bottom up. Here, we propose a conceptual algorithm which is instead divisive, starting from the cluster containing all points and building a hierarchy of elevated subsets in descending fashion by recursively splitting out clusters.

Our algorithm takes as input a $n \times n$ similarity space produced by a similarity function S, such that if data points x_1 and x_2 are more similar than data points x_3 and x_4, then element $S[1, 2]$ in our similarity matrix is greater than element $S[3, 4]$. In the similarity space, the upper-triangular region includes all of the similarities of the largest subset possible in the data - that subset consisting of the similarities of the entire dataset. If we name this maximal subset A, $s(A)$ is the minimum similarity between any pair of points in the entire dataset. It follows then that any subset of this A can not contain this pair of points and have a higher s-level, and therefore, no subset containing this pair can be an elevated subset.

With this insight, we need only generate $(n-1)$-sized subsets of A that do not contain the least similar pair of points. Thus, instead of searching through $\binom{n}{n-1}$ subsets, we only generate 2, each one leaving out one member of the least similar pair of points. These two descendant subsets are either elevated or equal in s-level to A.

The s-level of each of these resulting subsets will again be due to a particular pair of points, and any descendant subset containing that pair can not be an elevated subset. So, each size $(n-1)$ subset produces only two size $(n-2)$ subsets - each one leaving out one point of the minimum similarity pair.

This process continues down to the size $(n-(n-1))$ subsets, on the kth step generating at most 2^k potential elevated subsets. In practice, the actual number of candidate subsets at each level is much lower due to particular points (outliers) showing up repeatedly as members of the least-similar pair in several candidates at a particular level. A simple check at each level removes repeated candidate subsets, often dramatically reducing the branching factor.

The *rise* for a particular elevated subset is easily computed as the difference between a subset's s-level and that subset's parent's s-level. This is true because at each level, a subset has an s-level either equal to or greater than its parent, meaning that a subset's parent has the highest s-level in its ancestry. Consequently, a subset's rise is simply the difference in s-level between the subset and its parent.

Cluster Selection

As in hierarchical agglomerative clustering, the hierarchy of elevated sets does not yield a clustering of the data, but instead an enumeration of potential clusters. Identifying clusters requires a pass through the identified elevated sets where clusters are selected to represent the data based on their merit according to some objective function. Such an objective function should capture what is meant by a “good” cluster. The intuitive idea of a good cluster is a specific set of points that is dense relative to a set of points drawn randomly from the data. *Density* in a similarity space, henceforth “s-density”, is the mean of all pairwise similarities among points in the cluster, with a high s-density indicating a dense cluster.

The exhaustive top down algorithm described above finds at each level the data's elevated subsets of a particular size. For some sizes, the s-density of the elevated subsets may only minimally exceed that of the background, while in others, the elevated subsets' s-density may be significantly higher than the background. Elevated subsets exhibiting a high s-density relative to the background likely represent meaningful clusters. In order to find such clusters, we can compute the relative s-density of elevated subsets versus random subsets of the same size by comparing a p-value of each elevated subset's s-density against a normal distribution of s-densities of randomly-sampled subsets. Elevated subsets with a p-value below some threshold would be flagged as good clusters.

The procedure for selecting clusters from elevated subsets of size g given a significance threshold of p_{sig} is as follows:

1. Sample q random subsets of size g from the data,
2. Compute the mean and variance of s-densities over all of the random subsets,

3. For each of the discovered elevated subsets, compute the normal p-value of the subset's s-density over a normal distribution with the mean and variance computed in (2),
4. Flag subsets with p-values less than p_{sig} as clusters, proceed to next level.

6.5.2 Feasible Algorithm for Enumerating Elevated Subsets

The elevated subset enumeration algorithm and cluster selection algorithm described above will exhaustively identify all convex, multiple membership clusters in a similarity space which meet the significance criteria p_{sig}. The enumeration algorithm, however, requires computational time and memory on the order of $n \cdot 2^n$, which is infeasible for any realistic-sized dataset. While exhaustively examining the similarity space is infeasible, this section describes appropriate heuristics intended to modify the exhaustive algorithm into a feasible, if not exhaustive, form.

In Figure 6.8, it is notable that all nodes at the apex and base of the multiple-membership dendrogram are also nodes of the hierarchical agglomerative single-membership dendrogram. For the data that the dendrogram represents (Table 6.3), the base nodes are singleton clusters and the apex is the cluster containing all points. However, the base points could easily be complete-link clusters containing several points, and the apex could be a complete-link cluster containing a subset of the entire dataset. In general, as long as the apex and base clusters are convex, they can contain any number of points. Therefore, Figure 6.8 could represent the enumeration of elevated subsets between any two levels of a single-membership hierarchical agglomerative nested cluster enumeration.

This observation indicates a method for dramatically pruning the search space of the exact elevated subset enumeration algorithm described above: for a specified maximum and minimum similarity level, elevated subsets need only be enumerated between those levels on a single-membership tree created by single-link hierarchical agglomerative clustering. Thus, only hierarchical agglomerative, which due to its greedy nature is a faster algorithm, needs to examine the whole similarity space; then the slower elevated subset enumeration need only proceed over a small region of the discovered single-link dendrogram. However, even on a limited region of the search space, elevated subset enumeration remains expensive due to each descending step's 2^k branching factor. This branching factor can be ameliorated to some degree by applying the cluster selection algorithm in an online fashion, selecting clusters at each iteration and only proceeding to generate descendants of those clusters not selected.

Even with these modifications, the algorithm's computational expense may negate its utility in most situations. A practical solution is to minimize the generation of potential subsets, and instead to focus on the intuition of bounding the search space using s-levels in the hierarchical agglomerative tree. As is clear from Figure 6.8, for any given lower and upper bound on the s-level, the clusters at the upper bound are the fine, elemental convex subsets that are merged to form the larger, coarser clusters at the lower bound. The coarse clusters do not contain overlapping points because the local merges performed by the hierarchical agglomerative algorithm do not allow multiple membership. However, the preceding discussion demonstrates that had the set of elevated subsets been enumerated, the coarse, lower s-level subsets would contain blocks of overlap corresponding to the fine convex sets of the upper

s-level bound. This observation suggests a simple algorithm for enumerating elevated subsets at the upper similarity bound:

1. For a given upper and lower bound on s-level, find the set of coarse subsets C with s-level greater than the lower bound and a set of fine subsets F with s-level greater than the upper bound where each element of C consists of a set constructed from a subset of the elements of F.
2. For each coarse subset C_i,
    (a) identify the constituent fine subsets of C_i,
    (b) for each fine subset F_j not a constituent of C_i, if the subset $C_i \bigcup F_j$ has an s-level greater than the lower bound, add $Ci \bigcup F_j$ to the set C.
3. Repeat 2 until no new clusters are added to C.
4. Eliminate duplicates from C and return the list of clusters.

The resulting set C will be overlapping clusters meeting the criteria of the lower similarity bound. While it may be unreasonable to ask a user to define a lower bound on similarity in discovered clusters, the value of such a bound can be easily computed by existing methods of cluster selection in hierarchical trees including stability and number of clusters.

6.6 Similarity Space Density-Searching Multiple Membership Algorithms

As mentioned previously, it is convenient to conceptualize a similarity space as a weighted graph with n nodes where the measured similarity $S[i, j]$

between elements i and j forms an arc of weight $S[i, j]$ between the vertex representing i and the vertex representing j. In this representation, we can define the *neighborhood* of a test point as the set of points connected to the test point via arcs with weight values greater than a given threshold t. If the arc weights are thought of as distances, a neighborhood can be visualized as the points within a sphere of radius r around the test point, with r inversely proportional to t. The *density* of a neighborhood is then the ratio of the number of points contained in the sphere of radius r to the magnitude of r. The problem of clustering in a similarity space is finding those regions of highest relative density, and collecting them in a concise and understandable way.

Elevated subsets, as described in Section 6.4.1, are the convex regions of density in a similarity space. Enumerating them is expensive, and the enumeration technique depends upon the complete-link property which ensures convexity in the discovered densities. In this section, we describe a density hill-climbing search algorithm which, instead of enumerating, uses a greedy search to identify elevated subsets likely to be clusters. Additionally, this algorithm is parameterized such that the complete-link requirement can be relaxed, allowing the discovery of non-convex regions of density.

Although it is a model-based, the k-means algorithm's iterative reassignment is a good example of hill-climbing: on each iteration, each of the k centers is relocated to a region of higher density. While k-means' density-climbing does not guarantee a good solution, the requirement that each data point is assigned to one center prevents a degenerate solution where all points are in the same cluster. Conversely, a technique that allows multiple-membership must take specific consideration to prevent degeneracy. Our approach prevents degeneracy by defining two zones around each point: a core and a periphery.

These two zones are defined by the similarity thresholds t_{core} and $t_{periphery}$, where $t_{periphery} < t_{core}$. With these two thresholds defined, each point has one of three possible relationships with other points in the data set: points similarities to the test point greater than t_{core} are *C-neighbors*, points with similarities greater than $t_{periphery}$ but less than t_{core} are *P-neighbors*, and points with similarities less than $t_{periphery}$ are *ϕ-neighbors* (not neighbors at all). After application of the thresholds to the similarity space matrix, the similarity space matrix is discretized such that each element is a C, P, or a ϕ corresponding to the neighborhood assignment of each point relative to every other point. Figure 6.9 illustrates these relationships.

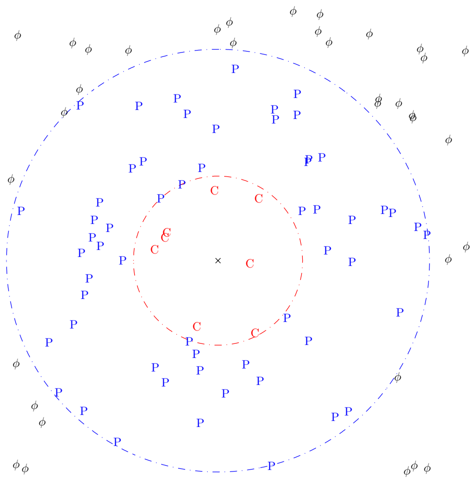

Figure 6.9: For the point at the center of the figure, the sphere representing the core threshold t_{core} is shown in red and the sphere representing the peripheral threshold $t_{periphery}$ is shown in blue. The points composing the *C-neighborhood* are shown in red, those of the *P-neighborhood* in blue, and those of the *ϕ-neighborhood* in black.

With the similarity space discretized as described, the algorithm proceeds by repetitive application of three fundamental procedures: cluster core finding, cluster growth, and cluster merging.

6.6.1 Cluster Core Finding

A "cluster core" is a C-neighborhood which forms the core of a developing cluster. Desirable cores have high density, which is measured as the tally of C-neighbors. When several candidate points are available to be cores, selecting the best involves tallying each point's C-neighbors and ranking the points according to both the tally and to rules regarding sharing of points between clusters.

6.6.2 Cluster Growth

C-neighborhoods, which form the core of clusters, are convex regions of density; however, in reality clusters may be arbitrarily shaped. Thus, after the selection of convex cluster cores, there is a need to allow the cluster to adjust its shape to fit the data. The mechanism of reshaping by cluster growth calls upon the graph conceptualization of the similarity space and the idea of message passing between points. Upon selection of a cluster core, the contained points (*C-neighbors* of the central point) are assigned cluster membership. Peripheral points (*P-neighbors*) are evaluated for addition to the cluster depending on their *connectivity* to the cluster members. *Connectivity* is defined as the fraction of cluster members which are *C-neighbors* to the peripheral point. Restated in the message passing conceptualization, if each point contained within the cluster sends a message only to its *C-neighbors*, a peripheral point's connectivity is the number of messages it receives divided by the total number of cluster members. Figure 6.10 illustrates *P-neighbor* connectivity. On a single iteration, points with connectivity exceeding the connectivity threshold are incorporated into the cluster. Iterations proceed until no new points are added.

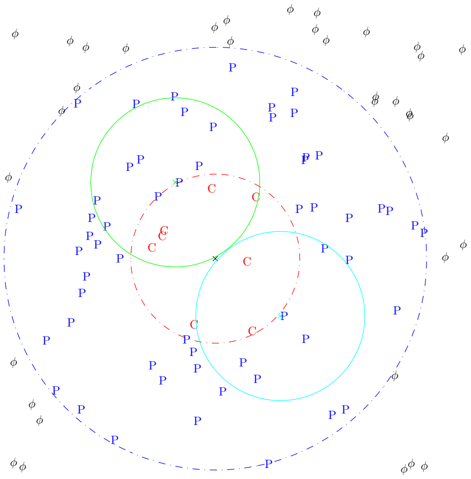

Figure 6.10: Criteria for granting *P-neighbors* cluster membership. The point indicated in green is in the *P-neighborhood* of the point at the center of the cluster, and contains more than half of the cluster's members in its own *C-neighborhood*, therefore, it would be added to the cluster. The point indicated in cyan is also in the *P-neighborhood* of the cluster, but contains very few of the cluster's members in its *C-neighborhood* and therefore would not be added to the cluster.

6.6.2.1 Connectivity and Single-Link *vs* Complete-Link Character

The choice of connectivity threshold determines the shape of similarity space clusters. Connectivity thresholds closer to zero give the cluster growth algorithm a single-link character, allowing chaining growth along paths of density. Connectivity thresholds closer to one give the algorithm a complete-link character, resulting in more spherical clusters.

6.6.3 Cluster Merging

As clusters expand according to the cluster-growth procedure, the lack of the single-membership constraint allows them to absorb points already assigned to other clusters. Eventually, clusters may share so many points that they parsimoniously represent one similarity density rather than two. This condition is defined by a *merge threshold*, calculated as the fraction of cluster members which are shared with another cluster. If two clusters mutually exceed the merge threshold with each other, they are merged into one and their constituent points all given the same label.

6.6.4 Parameters

Including the thresholds needed for similarity space discretization, these base procedures require five user-defined parameters:

Parameter	Meaning
C-Neighbor Threshold	the minimum similarity required to be a *C-neighbor*
P-Neighbor Threshold	the minimum similarity required to be a *P-neighbor*
Minimum Cluster Size	the minimum number of points required to form a cluster
C-Neighbor Connectivity	the number of cluster members that must be *C-neighbors* for a point to be added to the cluster
Merge Threshold	the maximum fraction of points a cluster may share with another cluster

Table 6.4: Parameters for similarity space density hill-climbing algorithm.

6.6.5 Multiple-Membership Density Hill Climbing Algorithm

With a discretized similarity space and the parameters in Table 6.6.4 defined, the multiple-membership density climbing algorithm proceeds as follows:

1. Choose a point randomly to be a cluster seed.
2. Find the seed's *C-neighborhood* and *P-neighborhood.*
3. Apply the "Cluster Growth" procedure the cluster until it stabilizes.
4. Apply the "Cluster-Core Finding" procedure to the members of the cluster. If a member of the mature cluster has a denser *C-neighborhood* and that point is not in an existing cluster's *P-neighborhood*, relocate the seed to that point and repeat 2-3.
5. Continue relocating and growing until no points in the cluster have a greater density than the center.
6. Evaluate this cluster's overlap with existing clusters; if overlap exceeds *Merge Threshold*, merge the clusters.
7. Repeat 1-6 until k clusters are recovered, or no points meet the criteria to be a seed.

6.7 Experiments

6.7.1 Strawman for Comparison

To evaluate the performance of our multiple-membership density algorithm, we compare with a conceptually simple "strawman" algorithm. In similarity space, algorithms which require data in a metric space are not applicable, thus the previously described method of employing expectation maximization followed by threshold selection is not usable. The hierarchical agglomerative method requires only the similarities between points; however, it is inherently a single-membership method. In order to make the clusterings generated by hierarchical clustering comparable to the clusterings generated by the overlapping methods, we make some modifications to the cluster selection process following the construction of a hierarchical clustering. We select clusters from the hierarchical tree based on three criteria: internal similarity, size, and stability. We first identify all potential clusters in the hierarchical tree which meet both the criteria for minimum similarity of the *C-Neighbor Threshold* from Table 6.6.4 and minimum size (*Minimum Cluster Size* from Table 6.6.4). The resulting potential clusters will include several which are nested sequences of related clusters. When this occurs, we select the member of the nested sequence with the highest stability, where stability is measured as the change in internal similarity when the cluster is merged.

After choosing representative clusters in this manner, we have a non-overlapping, convex clustering consisting of clusters with a size $\geq$ *Minimum Cluster Size* and a similarity $\geq$ *C-Neighbor Threshold*. To give this clustering some overlapping character, we add to each cluster any points which are within a similarity of at least *P-Neighbor Threshold* of at least *C-Neighbor Connectivity* of the cluster members. In essence, we run one iteration of the "Cluster Growth" procedure described in Section 6.6.2. As this step can significantly alter the membership of the clusters, as a final step we evaluate the resulting memberships for potential merges using the "Merge" procedure described in Section 6.6.3 with the parameter *Merge Threshold* described in Table 6.6.4.

6.7.2 Datasets

For our experiments, we employed three diverse synthetic datasets generated using the model described in Section 6.2.2. The details of these datasets are given in Appendix B.

6.7.3 Results

Both our algorithm and the strawman algorithm have several tunable parameters and it is likely that careful parameter selection can improve their performance; however, in these experiments we have aimed to simulate a "real-world" scenario of data analysis with no *a priori* information about optimal parameters. Therefore, we have chosen parameters which indicate no prior knowledge of the statistics of the similarity space. Table 6.7.3 lists the parameters used in the experiments.

Parameter	value
C-Neighbor Threshold	0.66
P-Neighbor Threshold	0.33
Minimum Cluster Size	10
C-Neighbor Connectivity	0.9
Merge Threshold	0.75

Table 6.5: Parameters used for experiments.

	Strawman		Density Climbing	
Data	Clusters	Omega	Clusters	Omega
Synthetic 1	21 ± 0	0.494 ± 0	11.9 ± 0.876	**0.674 ± 0.0238**
Synthetic 2	18 ± 0	0.412 ± 0	15.3 ± 1.42	**0.481 ± 0.00853**
Synthetic 3	19 ± 0	0.432 ± 0	13.9 ± 0.994	**0.554 ± 0.0182**

Table 6.6: Results of algorithms compared to strawman on synthetic datasets. Because of its random initialization, results for the Density Climbing algorithm are averaged over ten runs.

6.7.4 Similarity Space Density Climbing Applied to Microarray Data

In microarray experiments, it is generally accepted that genes that show high absolute correlation across a series of experiments are together involved in an underlying biological process. Finding such highly-correlated groups of genes identifies functional blocks in cellular machinery. Individual genes, however, often play roles in several different processes, and should be grouped accordingly into all appropriate functional groups. The problem of clustering genes into process groups based on their expression across experiments is a natural application of overlapping clustering.

In this section we apply the similarity space density climbing algorithm to the Gasch gene expression dataset[GE02], specifically using the subset of highly active genes identified by Segal in [SBK03a]. The dataset consists of the expression levels of 1010 genes across 173 experimental conditions. We construct the similarity space by measuring the absolute value of the Pearson correlation coefficient between each gene's expression vector. We then apply the similarity space density climbing algorithm with the following parameters:

Parameter	value
C-Neighbor Threshold	0.5
P-Neighbor Threshold	0.4
Minimum Cluster Size	10
C-Neighbor Connectivity	0.9
Merge Threshold	0.75

Over ten runs of the algorithm, the average number of clusters discovered was 15.1±2.1. Distributions of within-cluster similarity versus background similarity were consistent across the runs, with an example set of distributions shown in Figure 6.11.

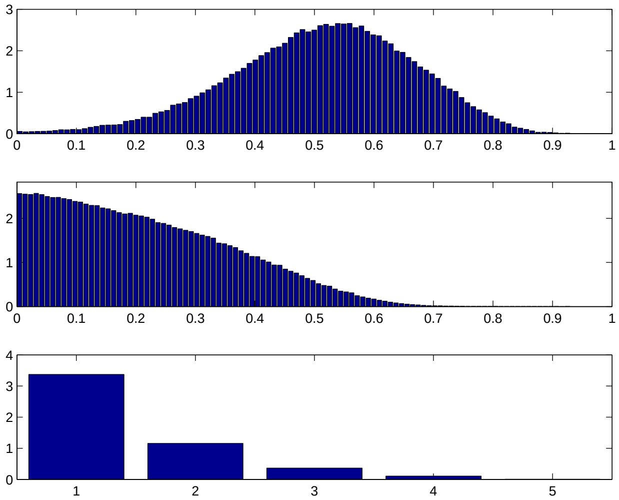

Figure 6.11: Distributions of within-cluster similarity (top pane), between-cluster similarity (middle pane), and shared labels (bottom pane) of the clustering found by the similarity space density hill climbing algorithm. It is notable by comparing the top and middle panes that the discovered clusters have a much higher internal similarity than the background. The bottom pane shows that genes in this clustering belong to as many as 5 different clusters.

Figure 6.11 demonstrates that the algorithm has appropriately discovered sets of high internal similarity in the similarity space. We can evaluate the biological meaning of these high-similarity groupings by looking for enrichment of process labels among genes which have been annotated. We calculate the enrichment of a particular cluster as the hypergeometric p-value of its most frequent label relative to the frequency of that label in the entire set of genes. Examining specific clusters from the clustering displayed in Figure 6.11, we find:

Cluster	Process Name	Enrichment
1	generation of precursor metabolites and energy	1.73e-05
2	electron transport	1.46e-04
3	vecicle mediated transport	2.68e-03
4	carbohydrate metabolism	2.97e-04
5	generation of precursor metabolites and energy	7.71e-03
6	cytoskeleton organization and biogenesis	4.71e-05
7	amino acid and derivative metabolism	2.96e-08
8	amino acid and derivative metabolism	1.44e-10
9	generation of precursor metabolites and energy	1.32e-09
10	amino acid and derivative metabolism	6.43e-11
11	nuclear organization and biogenesis	2.17e-06
12	protein catabolism	8.92e-06

The high enrichments of cellular processes in these discovered clusters indicates that the discovered overlapping clusters correspond to sets of genes involved in biochemical processes.

6.7.5 Discussion

Table 6.6 demonstrates that for datasets fitting the structure described in Section 6.2, the proposed Density Climbing algorithm finds clusters which more closely match the ground-truth latent structure in the data than those found by the strawman algorithm. The demonstration in Section 6.7.4 shows the utility of the approach on the real-world overlapping clustering problem of identifying functional groups of genes from their expression profiles in collections of microarray experiments.

Chapter 7

Conclusions and Future Directions

7.1 Contributions

The idea of overlapping clustering is not new, as demonstrated by the catalogue of existing algorithms given in Chapter 2, however multiple membership techniques have largely remained outside the armamentarium of modern data analysts. Prominent among the reasons for this are three issues: first, that the existing algorithms have been largely developed *ad hoc* for particular tasks, with little effort toward generalization; second, that neither the purpose nor meaning of allowing multiple memberships in clusters has been clear; and third, that without general-purpose multiple-membership algorithms and tools for analysis, the richer but more challenging-to-interpret information yield from overlapping methods has not exceeded the simpler but well-understood yields of single-membership methods. This dissertation has addressed each of these issues, with the aim of increasing the understanding of multiple-membership clustering and providing tools to make overlapping approaches more theoretically sound, easier to use, and easier to analyze.

Chapter 1 clarified clustering's role as a technological tool much like telescopes or microscopes in allowing humans to see further than our innate senses allow, and framed overlapping cluster as a mechanism for communicating more information about structure in data than single-membership methods, with little increase in complexity. Chapter 2 catalogued existing approaches to overlapping cluster. Chapter 3 more closely examined the clustering-as-communication concept introduced in Chapter 1 while reviewing existing multiple-membership analysis tools and proposing new ones. Chapter 4 looked at a particular problem in overlapping clustering analysis: aligning clusters from different algorithms so as to make their meanings interpretable and comparable. Chapter 5 provided a general, model-based technique for analyzing data resulting from additive overlapping cluster. Chapter 6 addressed overlapping clustering in similarity spaces, proposing a generative model of similarity space overlapping clustering as well as algorithmic approaches to finding overlapping clusters in such spaces.

7.2 Future Work

As overlapping clustering is a still-developing field, there are several avenues for future development. Notable among these are techniques for visualization and interpretation, new algorithms and new means of comparison, and techniques for model selection.

Given clustering's role in guiding data analysis and interpretation, effectively and efficiently communicating clustering results to human users is critical. The proverb "a picture is worth a thousand words" holds particular validity when translating the results of high-dimensional numerical analysis into human-interpretable form. This dissertation presented some novel forms of visualization, notably the *cluster signatures* presented in Chapter 4 and the multiple-membership dendrogram of Figure 6.8; however, the need for more and better presentation methods is clear.

The algorithms described in Chapters 5 and 6 provide multiple-membership generalizations of the two primary representatives of single-membership clustering: model-based global optimization like k-means, and locally greedy models such as hierarchical agglomerative. Clearly, there are numerous other single-membership approaches which may generalize well to multiple-membership applications, and the work of identifying these approaches and their generalizations will provide research fodder into the forseeable future.

Chapter 6 gives a very general model of, and algorithms for analyzing, the structure of data containing overlapping clusters. A consequence of the model's and algorithms' generality is the large number of parameters required. While the experiments demonstrate that even naive choices of these parameters yield good results, a means of identifying optimal parameters *a priori* would greatly enhance the algorithms' utility. This model selection problem is nontrivial - it is still a prominent issue in single-membership clustering. However, the similarity-space generative model presented in Chapter 6 provides a means of performing experiments intended to correlate the observable structure of a dataset (e.g. histograms, heatmaps) with the latent structure sought by the cluster-finding algorithms. Exploration of these correlations may yield model selection guidance for not only multiple-membership algorithms, but single-membership as well.

Appendices

Appendix A

Mutual Information when labels exceed datapoints

Consider the situation where we have 2^k labels and N datapoints, where $N < 2^k$. If we assume that each data point is assigned one of the 2^k labels with equal probability, than with high probability, each point will have a different label. For two clusterers assigning points in this manner, the joint probability between a label in clusterer 1 and a label in clusterer 2 will be:

$$p(\text{label}_1, \text{label}_2) = \frac{1}{N}$$

and any other joint probability involving label_1 and label_2 will be zero.

Since label_1 has occurred once in the N data points, its prior will be:

$$p(\text{label}_1) = \frac{1}{N}$$

as will the prior of label_2.

So, the mutual information calculation will be:

$$\begin{aligned} \text{MI} &= \sum_{i=1}^{N} p(\text{label}_{i1}, \text{label}_{i2}) \log_2 \frac{p(\text{label}_{i1}, \text{label}_{i2})}{p(\text{label}_{i1}) p(\text{label}_{i2})} \\ &= \sum_{i=1}^{N} \frac{1}{N} \log_2 \frac{\frac{1}{N}}{(\frac{1}{N})(\frac{1}{N})} \\ &= \log_2 N \end{aligned}$$

which is clearly nonzero.

Appendix B

Synthetic Datasets

This section describes the three synthetic datasets generated using the generative model described in Chapter 6 and employed in the evaluation of the algorithms described in that chapter.

B.1 Synthetic Dataset 1

Dataset 1, previously described in Chapter 6, is intended to be characteristic of a real-world similarity space containing clusters. Table B.1 lists the parameters used to generate this dataset. The 10 clusters are sized between 10% and 20% of the size of the data, share around 20% of their points, and have a mixed single-link/complete-link character. Figure B.1 illustrates this dataset's within-cluster and between-cluster histograms.

Parameter	Value
n	500
k	10
Q	discrete Uniform over $(\lfloor \frac{n}{10} \rfloor, \lfloor \frac{n}{5} \rfloor)$
c	0.8
v	0.2
$\Theta_{background}$	(1.5,5)
$\Theta_{cluster}$	(10,3)

Table B.1: Parameters of Synthetic Dataset 1

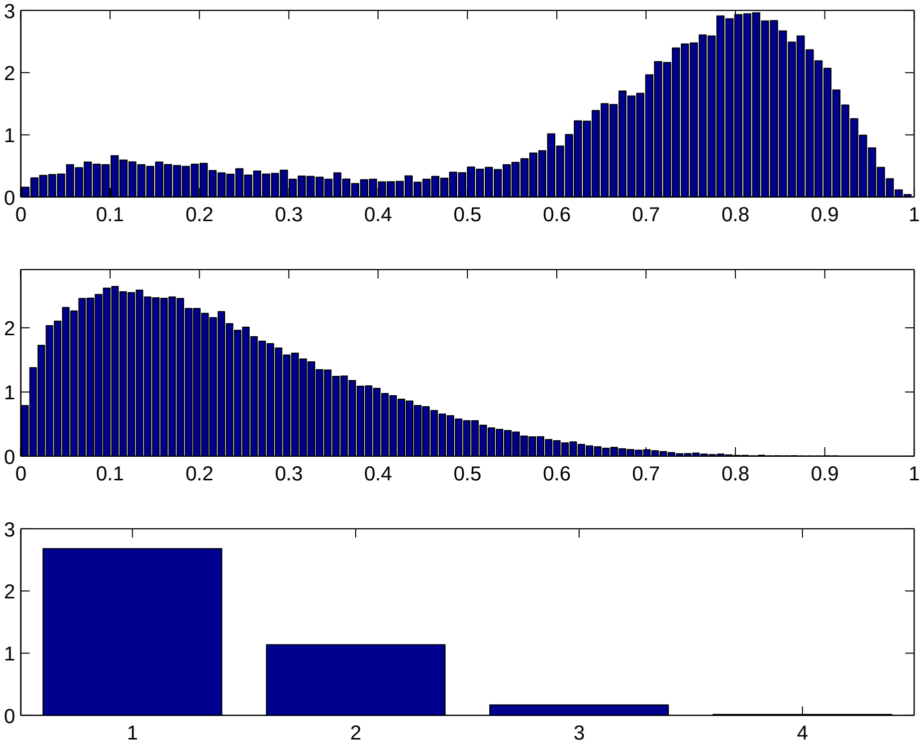

Figure B.1: Characteristics of the synthetic data set generated using the parameters in Table B.1. The first pane shows the distribution of within-cluster similarities, the second shows the background similarity distribution, and the bottom pane the distribution of memberships. Note that the within-cluster distribution is a mixture of the specified within-cluster Beta distribution and a small amount of the background Beta distribution. This is due to the parameter c which specifies complete-link character being less than one: points within a cluster are not required to have high similarity to all other points within the cluster.

Parameter	Value
n	500
k	30
Q	discrete Uniform over $(\lfloor \frac{n}{500} \rfloor, \lfloor \frac{n}{100} \rfloor)$
c	0.5
v	0.3
$\Theta_{background}$	(1.5,5)
$\Theta_{cluster}$	(5,1.5)

Table B.2: Parameters of Synthetic Dataset 2

B.2 Synthetic Dataset 2

Dataset 2 consists of dense, nonconvex clusters with significant (30%) overlap. This dataset is intended to be representative of the similarity structure of a social network, with several subsets of nodes shared between several larger networks. Table B.2 lists the parameters used to generate this dataset, and Figure B.2 illustrates this dataset's within-cluster and between-cluster histograms.

B.3 Synthetic Dataset 3

Dataset 3 consists of nonconvex clusters with minimal (5%) overlap. Table B.3 lists the parameters used to generate this dataset, and Figure B.3 illustrates this dataset's within-cluster and between-cluster histograms.

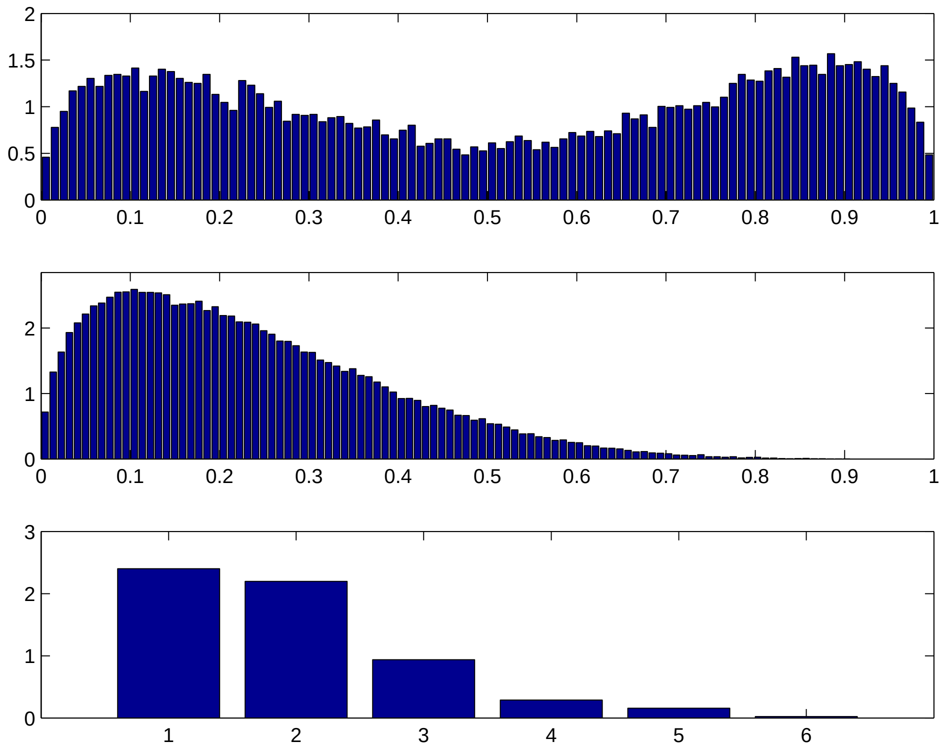

Figure B.2: Structure of Synthetic Dataset 2. The top pane shows the within-cluster distribution of similarities, the middle pane shows the distribution of background similarities, and the bottom pane the distribution over number of cluster memberships per point.

Parameter	Value
n	500
k	10
Q	discrete Uniform over $(\lfloor \frac{n}{500} \rfloor, \lfloor \frac{n}{100} \rfloor)$
c	0.5
v	0.05
$\Theta_{background}$	(1.5,5)
$\Theta_{cluster}$	(15,5)

Table B.3: Parameters of Synthetic Dataset 3

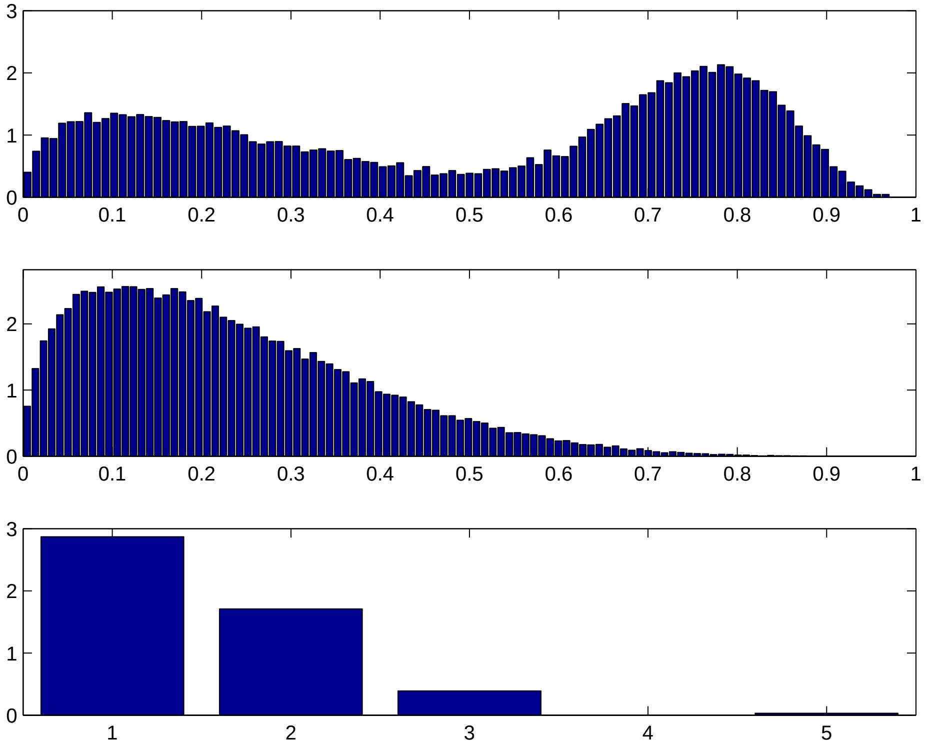

Figure B.3: Structure of Synthetic Dataset 3. The top pane shows the within-cluster distribution of similarities, the middle pane shows the distribution of background similarities, and the bottom pane the distribution over number of cluster memberships per point.

Bibliography

[ABB00] O. Alter, P.O. Brown, and D. Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, 97:10101–10106, August 2000.

[ACDW81] Phipps Arabie, J. Douglas Carroll, Wayne DeSarbo, and Jerry Wind. Overlapping clustering: A new method for product positioning. *Journal of Marketing Research*, 18(3):310–317, 1981.

[ADLCJ99] J.C. Aude, Y. Diaz-Lazcoz, J.J. Codani, and J.L.Risler. Applications of the pyramidal clustering method to biological objects. *Computers and Chemistry*, 23:303–315, 1999.

[Ara77] P. Arabie. Clustering representations of group overlap. *Journal of Mathematical Sociology*, 5:113–128, 1977.

[AS73] P. Arabie and R. N. Shepard. Representation of similarities as additive combinations of discrete, overlapping properties. *Psychological Review*, 86(2):87–123, 1973.

[BBK$^+$05] Arindam Banerjee, Sugato Basu, Chase Krumpelman, Joydeep Ghosh, and Raymond Mooney. Model based overlapping clustering. In *Proceedings of KDD2005*, pages 100–106, 2005.

[BBM04] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD*, 2004.

[BD85] P. Bertrand and E. Diday. A visual representation of the compatibility between an order and a dissimilarity index. *Computational Statistics Quarterly*, 2:31–42, 1985.

[Bez81] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, 1981.

[BHB04] Gill Bejerano, David Haussler, and Mathieu Blanchette. Into the heart of darkness: Large scale clustering of human non-coding dna. *Bioinformatics*, 20:i40–i48, 2004.

[Bjo96] A. Bjorck. *Numerical Methods for Least Squares Problems*. Society for Industrial & Applied Math (SIAM), 1996.

[BMDG04] Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. In *SDM*, 2004.

[BTGM04] Jean-Phillipe Brunet, Pablo Tomayo, Todd R. Golub, and Jill P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences*, 101(12):4164–4169, 2004.

[CC00] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proc. 8th Intl. Conf. on Intelligent Systems for Molecular Biology (ICMB)*, pages 93–103, 2000.

[CD88] Linda M. Collins and Clyde W. Dent. Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions. *Multivariate Behavioral Research*, 23(2):203–230, 1988.

[Chv80] V. Chvátal. Hard knapsack problems. *Operations Research*, 28(6):1402–1412, 1980.

[Con66] P. Constantinescu. The classification of a set of elements with respect to a set of properties. *Computer Journal*, 8:352–357, 1966.

[CT91] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[DDL$^+$90] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[DK03] Doulaye Dembele and Philippe Kastner. Fuzzy c-means method for clustering microarray data. *Bioinformatics*, 19(8):973–980, 2003.

[DLR77] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[DM01] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42:143–175, 2001.

[ea00] A. A. Alizadeh et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–510, 2000.

[FDD$^+$88] George W. Furnas, Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, Richard A. Harshman, Lynn A. Streeter, and Karen E. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of SIGIR 1988*, pages 465–480, 1988.

[FGKP99] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, 1999.

[FPK01] U. Feige, D. Peleg, and G. Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[GE02] Audrey P. Gasch and Michael B. Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biology*, 3:0059.1–0059.22, 2002.

[GG05] T. L. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. Technical Report 2005-001, Gatsby Computational Neuroscience Unit, 2005.

[GI89] D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms.* MIT Press, Cambridge, MA, 1989.

[GI91] Zvi Galil and Guiseppe F. Italiano. Maintaining biconnected components of dynamic planar graphs. In *Proc. 18th Intl. Colloquium on Automata, Languages, and Programming*, pages 339–350. Springer-Verlag, Berlin, 1991.

[GKT05] David Gibson, Ravi Kumar, and Andrew Tomkins. Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st VLDB Conference, Trondheim, Norway*, 2005.

[GL96] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, 1996.

[GS62] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematics Monthly*, 69:9–14, 1962.

[GSK$^+$00] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Stotz, D. Botstein, and P.O. Brown. Genomic expression program in the response of yeast cells to environmental changes. *Molecular Cell Biology*, 11:4241–4257, 2000.

[HA85] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[Har69] Frank Harary. *Graph Theory.* Addison-Wesley, Reading, Mass., 1969.

[Har75] John A. Hartigan. *Clustering Algorithms.* John Wiley & Sons, 1975.

[Hoj83] Hiroshi Hojo. A maximum likelihood method for additive clustering and its applications. *Japanese Psychological Research*, 25(4):191–201, 1983.

[HS00] Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76:175–181, 2000.

[HTE$^+$00] Trevor Hastie, Robert Tibshirani, Michael B. Eisen, Ash Alizadeh, Ronald Levy, Louis Staudt, Wing C. Chan, David Botstein, and Patrick Brown. 'gene shaving' as a method for identifying distinct sets of genes with similar expression paterns. *Genome Biology*, 2:0003.1–0003.21, 2000.

[HW79] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.

[HYH$^+$05] Haiyan Hu, Xifeng Yan, Yu Huang, Jiawei Han, and Xianghong Jasmine Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21:i213–i221, 2005.

[JS68] N. Jardine and R. Sibson. The construction of hierarchic and non-hierarchic classifications. *Computer Journal*, 11:177–184, 1968.

[LD84] X. Li and R. C. Dubes. The selection of significant dichotomous features. In *Proceedings of the Seventh International Conference on Pattern Matching*, pages 260–264, 1984.

[LD85] X. Li and R. C. Dubes. The first stage in two-stage template matching. In *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI 7*, pages 700–707. 1985.

[LS99] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[LS00] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[LS01] D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Syposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.

[MSW72] W. T. McCormick, P. J. Schweitzer, and T. W. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20:993–1009, 1972.

[Nai88] John Naisbitt. *Megatrends: Ten New Directions Transforming Our Lives.* Grand Central Publishing, 1988.

[NG05] Daniel J. Navarro and Thomas L. Griffiths. Bayesian additive clustering. In *Proceedings of Adelaide Mental Life, 2*, June 2005.

[Ran71] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

[SA79] Roger N. Shepard and Phipps Arabie. Additive clustering: Represention of similarities as combinations of discrete overlapping properties. *Psychological Review*, 86(2):87–123, 1979.

[SBK03a] E. Segal, A. Battle, and D. Koller. Decomposing gene expression into cellular processes. In *Proc. 8th Pacific Symposium on Biocomputing (PSB), Kaua'i*, Jan 2003.

[SBK03b] E. Segal, A. Battle, and D. Koller. Decomposing gene expression into cellular processes. In *PSB*, 2003.

[SG02] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining partitionings. In *Proc. Conference on Artificial Intelligence (AAAI 2002)*, pages 93–98. IEEE, July 2002.

[SM] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905.

[Spa73] D. N. Sparks. Algorithm as 58: Euclidean cluster analysis. *Applied Statistics*, 22(1):126–130, 1973.

[Tan96] Joshua B. Tanenbaum. Learning the structure of similarity. In D. S. Touretzky M. C. Mozer and M. E. Hasselmo, editors, *Neural Information Processing Systems (NIPS) 8*, pages 3–9, 1996.

[XLG03] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 267–273, 2003.

[Yao03] Y. Y. Yao. Information-theoretic measures for knowledge discovery and data mining. In Karmeshu, editor, *Entropy Measures, Maximum Entropy Principle and Emerging Applications*, pages 115–136. Springer, 2003.

Vita

Chase Serhur Krumpelman was born in Lexington, Kentucky and earned his Bachelor of Science in Electrical Engineering from the University of Kentucky. An internship at Motorola shifted his interest from hardware to data analysis and data mining, leading to his joining Joydeep Ghosh's Intelligent Data Exploration and Analysis Laboratory at UT Austin. Chase's graduate career at UT has involved forays into industry with engineering positions with the search startup NeonYoyo and the hedge fund Akul Group, Inc. His academic interests have been focused on data analysis issues arising from high-throughput biologic data, as well as developing software tools and educational materials. Chase is currently pursuing his Doctor of Medicine degree at Baylor College of Medicine and plans a career integrating clinical care with continuing investigation into the computational analysis of biomedical data.

Permanent address: 4805 Eilers Avenue
Austin, Texas USA
email: chase.krumpelman@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†] LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.