

An Analysis of Recent Work on Clustering Algorithms

Daniel Fasulo*

Department of Computer Science & Engineering
Box 352350
University of Washington
Seattle, WA 98195

April 26, 1999

Department of Computer Science & Engineering
Technical Report # 01-03-02

Abstract

This paper describes four recent papers on clustering, each of which approaches the clustering problem from a different perspective and with different goals. It analyzes the strengths and weaknesses of each approach and describes how a user could decide which algorithm to use for a given clustering application. Finally, it concludes with ideas that could make the selection and use of clustering algorithms for data analysis less difficult.

1 Introduction

1.1 Informal Problem Definition

Clustering can be loosely defined as the process of organizing objects into groups whose members are similar in some way. There are two major styles of clustering: *partitioning* (often called *k-clustering*), in which every object is assigned to exactly one group, and *hierarchical clustering*, in which each group of size greater than one is in turn composed of smaller groups.

Both hierarchical clustering and *k-clustering* had been studied extensively by the mid-1970's, and comparatively little clustering research was carried out in the 1980's. In recent

*Email: Daniel.Fasulo@celera.com

years, however, the advent of World Wide Web search engines (and, specifically, the problem of organizing the large amount of data they produce) and the concept of “data mining” massive databases has lead to a renewal of interest in clustering algorithms.

The goal of this paper is to examine recent work in clustering. In particular, the focus is on clustering algorithms whose goal is to **identify groups of related items in an input set**. This is in contrast to certain “clustering-like” problems such as graph partitioning [33] or segmentation problems [23], which organize objects into groups but often have other objectives. As much as possible, this paper also emphasizes the fundamental ideas behind the construction of clusters and attempts to avoid the details of specific clustering applications, such as determining good measures for text document similarity.

The remainder of this section summarizes prior work in clustering and certain related problems, and briefly describes the issues involved in representing the input to clustering algorithms. Section 2 focuses on summarizing and independently critiquing a set of four recent papers on clustering. Finally, Section 3 compares the strengths and weaknesses of the various algorithms and suggests ways in which the positive aspects of each could be synthesized in future work.

For a more detailed introduction to clustering, see the works by Everitt [12], Rasmussen [29], Kaufman and Rousseeuw [21], Jain and Dubes [20], and Gordon [18]. In particular, the works by Everitt and Jain and Dubes provide clear and comprehensive coverage of the “classical” approaches to clustering.

1.2 k -clustering

In general, k -clustering algorithms take as input a set S of objects and an integer k , and output a partition of S into subsets S_1, S_2, \dots, S_k . By far the most common type of k -clustering algorithm is the *optimization algorithm*. Optimization algorithms typically assume that the elements of S are drawn from a d -dimensional metric space, usually \mathbb{R}^d , and define a cost function $c : \{X : X \subseteq S\} \rightarrow \mathbb{R}^+$ which associates a cost with each cluster. The goal of the algorithm is then to minimize $\sum_{i=1}^k c(S_i)$, the sum of the costs of the clusters.

The most well-known optimization criterion is the *sum-of-squares criterion*. Let \mathbf{x}_r^i be the r th element of S_i , $|S_i|$ be the number of elements in S_i , and $d(\mathbf{x}_r^i, \mathbf{x}_s^i)$ be the distance between \mathbf{x}_r^i and \mathbf{x}_s^i . The sum-of-squares criterion is then defined with the following cost function:

$$c(S_i) = \sum_{r=1}^{|S_i|} \sum_{s=1}^{|S_i|} (d(\mathbf{x}_r^i, \mathbf{x}_s^i))^2. \quad (1.1)$$

This definition of the clustering problem is known to be NP-hard [15]. Nevertheless, the k -means algorithm of MacQueen [24] is a popular clustering algorithm which uses the sum-of-squares criterion. The algorithm relies on the ability to calculate the centroid of each

cluster S_i , denoted $\bar{\mathbf{x}}^i$. Technically the algorithm optimizes using the cost function

$$c(S_i) = \sum_{r=1}^{|S_i|} d(\bar{\mathbf{x}}^i, \mathbf{x}_r^i), \quad (1.2)$$

which can be shown to produce the same results as (1.1) above. Thus the k -means algorithm produces the requisite set of k clusters, along with the centroid (often termed the *representative element*) for each.

Since many types of data do not belong to spaces in which the mean is defined, Kaufman and Rousseeuw [21] have developed a similar algorithm for what they term the “ k -medioids” problem. Their algorithm, called PAM (for “Partitioning Around Mediods”), relies on the ability to find the median of each S_i , denoted $\hat{\mathbf{x}}^i$. Note that $\hat{\mathbf{x}}^i \in S_i$, and $\hat{\mathbf{x}}^i$ is chosen to minimize $\sum_{r=1}^{|S_i|} d(\hat{\mathbf{x}}^i, \mathbf{x}_r^i)$. This leads to the optimization criterion

$$c(S_i) = \sum_{r=1}^{|S_i|} d(\hat{\mathbf{x}}^i, \mathbf{x}_r^i). \quad (1.3)$$

Kaufman and Rousseeuw’s work has spawned a recent series of papers in the data mining community [28, 11, 10, 39, 1].

An alternate optimization criterion has been proposed by Gonzalez [17]. Instead of minimizing $\sum_{i=1}^k c(S_i)$, Gonzalez minimizes $\max_{1 \leq i \leq k} c(S_i)$, where $c(S_i)$ is given by

$$c(S_i) = \max_{\mathbf{x}_r^i, \mathbf{x}_s^i \in S_i} d(\mathbf{x}_r^i, \mathbf{x}_s^i). \quad (1.4)$$

This formulation is interesting mainly because there is a simple 2-approximation algorithm for its solution, which Gonzalez shows is the best bound possible if $P \neq NP$. Subsequent work by Xiang [38] has shown this approximation algorithm to be effective in practice for color quantization.

These optimization algorithms have several notable weaknesses. The first is that they heavily favor spherical clusters. Secondly, they do not deal adequately with “noise”; i.e., elements of S which do not cluster naturally with any other elements. Banfield and Raftery [2] and Celeux and Govaert [4] both develop frameworks in the context of statistical *mixture models* for clustering which subsume the optimization models above and deal with these issues. Mixture models in general and Banfield and Raftery’s work in particular will be discussed in Section 2.1.

An alternative k -clustering approach which addresses the issues of cluster shape and noise is called *density based* clustering. The intuitive idea of these approaches is that clusters can be considered “densely populated areas” in the space containing S . These areas can have arbitrary shape and ideally are well separated from one another. Although density based methods are comparatively uncommon, recent papers in the data mining literature

by Ester *et. al.* [10] and Agrawal *et. al.* [1] have followed this approach. The latter paper will be discussed in detail in Section 2.4.

Finally, closely related to density based clustering is *graph theoretic* clustering. Define a *similarity graph* $G(S)$ for the set S as follows: let S be the vertices of $G(S)$, and given $\mathbf{x}_r, \mathbf{x}_s \in S$, let $\{\mathbf{x}_r, \mathbf{x}_s\}$ be an edge in $G(S)$ if and only if \mathbf{x}_r and \mathbf{x}_s are “similar” under some definition of similarity. Intuitively, if S has an obvious cluster structure then $G(S)$ should appear as a collection of vertex-disjoint cliques, each corresponding to a cluster. In practice, $G(S)$ will not fit this description perfectly, but ideally should have highly-connected components separated by small sets of edges. Ben-Dor and Yakhini [3] use this observation to design a performance test for their clustering algorithm, which will be discussed further in Section 2.3. Hartuv *et. al.* [19] also use the similarity graph notion. They repeatedly apply a minimum cut algorithm to form clusters. Additional discussion of the relationship between certain graph problems and clustering can be found in a paper by Matula [25] and in Jain and Dubes’ book.

1.3 Hierarchical Clustering

Hierarchical clustering is an appealing approach to problems in which the input set S does not have a single “obvious” partition into well-separated clusters. The goal of a hierarchical clustering algorithm is to produce a tree $T(S)$ in which the nodes represent subsets of S . In particular, S itself is at the root of the tree, the leaves comprise the individual elements of S , and internal nodes are defined as the union of their children. A path down a well-constructed tree should then visit sets of increasingly tightly-related elements, and any set of nodes such that every path from a leaf to the root hits exactly one element in the set can be considered a partition of S into clusters. Hence, given the tree, the user can conveniently trade off between the number of clusters and the compactness of each cluster.

There are two major types of hierarchical clustering algorithms. *Divisive* algorithms work by recursively partitioning S until singleton sets are achieved; *agglomerative* algorithms work by beginning with singleton sets and merging them until S is achieved. This section will focus on agglomerative methods, which are far more common.

Most agglomerative methods follow the same conceptual framework. First, the elements are placed into a list of singleton sets S_1, S_2, \dots, S_n . Then a cost function is used to find the pair of sets $\{S_i, S_j\}$ from the list which is “cheapest” to merge. Finally, S_i and S_j are removed from the list of sets and replaced with $S_i \cup S_j$. This process is repeated until there is only one set remaining. Hence, the major difference between agglomerative clustering algorithms is the definition of the cost of merging two sets S_i and S_j , which will be denoted $c(S_i, S_j)$. Figure 1 summarizes the most well-known cost functions and lists examples of algorithms that use each.

In addition to the methods in Figure 1, *Ward’s method* [36] is also frequently used. Ward’s method can be seen as the hierarchical clustering equivalent to the sum-of-squares

Method	Cost Function	Representative Algorithm
Single-link	$\min_{\mathbf{x}_i \in S_i, \mathbf{x}_j \in S_j} d(\mathbf{x}_i, \mathbf{x}_j)$	SLINK [32]
Average-link	$\frac{1}{ S_i S_j } \sum_{\mathbf{x}_i \in S_i} \sum_{\mathbf{x}_j \in S_j} d(\mathbf{x}_i, \mathbf{x}_j)$	Voorhees' method [35]
Complete-link	$\max_{\mathbf{x}_i \in S_i, \mathbf{x}_j \in S_j} d(\mathbf{x}_i, \mathbf{x}_j)$	CLINK [6]

Figure 1: Common hierarchical clustering models

criterion for k -clustering. As in the above methods, at each step of the algorithm there is a list of clusters S_1, \dots, S_n and the goal is to choose a pair to merge. As in the sum-of-squares method, the cost of each S_i is defined as

$$c(S_i) = \sum_{r=1}^{|S_i|} \sum_{s=1}^{|S_i|} (d(\mathbf{x}_r^i, \mathbf{x}_s^i))^2. \quad (1.5)$$

The cost of the current solution is then defined as $\sum_{i=1}^n c(S_i)$, and the pair of sets to merge is chosen to minimize the increase in the cost of the total solution. Also as in the sum-of-squares method, algorithms implementing Ward's method typically rely on the ability to calculate the centroid of each cluster.

Hierarchical methods suffer similar criticism to statistical k -clustering methods. The average-link, total-link, and Ward's methods tend to favor spherical clusters, while single-link clustering is more analogous to density based methods and can produce undesirably "elongated" clusters. Nevertheless, hierarchical methods are widely used, particularly in the document clustering community [5, 37, 35].

1.4 Input to Clustering Algorithms

Fundamentally, every clustering algorithm operates on a set of input vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ in a d -dimensional space. It is important to note that the input spaces from different clustering problems may have different mathematical properties, and that these properties can influence which clustering algorithms may be applied to the data. For example, methods which calculate centroids cannot be used with spaces whose dimensions which are not numerical, and the method of Gonzalez in Section 1.2 must be used with vectors from a metric space.

In practice, many clustering algorithms do not directly examine or manipulate the input vectors. For example, the algorithms of Kaufman and Rousseeuw all take as input a *similarity matrix* M in which entry M_{ij} represents the degree of similarity between \mathbf{x}_i and \mathbf{x}_j . This matrix is assumed to be pre-calculated in an application-specific step. Their book goes into great detail on how to calculate similarity coefficients for different types of data, including non-numerical data.

A closely related style of input is the *distance matrix* D , in which D_{ij} represents the distance between element i and element j . This distance relation must typically follow the definition for a distance metric on the input space. A major advantage of this method is its simplicity; the Euclidean or Manhattan distance metrics can be used on any problem with numerical vectors.

Lastly, some clustering methods such as single-link hierarchical clustering and reciprocal nearest neighbor clustering [9, 27] require as input only a list of the input vectors and a function which returns the nearest neighbor of any input. This method has the advantage that the input is not of size n^2 , which may be prohibitive for some applications.

2 Four Recent Papers on Clustering

The purpose of this section is to introduce four recently-developed approaches to clustering. Significant comparison of these methods is withheld until Section 3 so that the reader can become familiar with all methods before they are compared and criticized. There are many tradeoffs involved in the construction of a clustering algorithm, and it would be misleading to present only the best features of each algorithm and claim that they could be merged easily into a single new, superior approach.

2.1 Banfield and Raftery: Mixture Models

This section describes the paper “Model-Based Gaussian and Non-Gaussian Clustering” by Banfield and Raftery [2]. Their approach is based on statistical *mixture models* for clustering. The idea of such models is that the input vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ are observations from a set of k unknown distributions E_1, E_2, \dots, E_k . Suppose the density of an observation \mathbf{x}_r with respect to E_i is given by $f_i(\mathbf{x}_r | \theta)$ for some unknown set of parameters θ . Also, suppose for each \mathbf{x}_r and distribution E_i , τ_r^i represents the probability that \mathbf{x}_r belongs to E_i . Each input is constrained to belong to some distribution, so $\sum_{i=1}^k \tau_r^i = 1$. Given these definitions, the goal of the scheme is to find the parameters θ and τ (defined as the vector of all the τ_r^i 's) that maximize the likelihood

$$L(\theta, \tau) = \prod_{r=1}^n \sum_{i=1}^k \tau_r^i f_i(\mathbf{x}_r | \theta). \quad (2.1)$$

Banfield and Raftery restrict their work to a simpler model in which each input must belong to exactly one distribution. Their work discards the τ vector and instead uses the vector $\gamma = (\gamma_1, \gamma_2, \dots, \gamma_n)$ to represent the identifying labels for the observations, so $\gamma_r = i$ if \mathbf{x}_r comes from E_i . The goal then is to find the parameters θ and γ which maximize the likelihood

$$L(\theta, \gamma) = \prod_{r=1}^n f_{\gamma_r}(\mathbf{x}_r | \theta). \quad (2.2)$$

Previous work in mixture models [31, 34] have modeled the distributions as multivariate normal functions; in this case the unknown parameters θ are the mean vector and covariance matrix of each distribution. Let $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ denote the mean vector and covariance matrix respectively for E_i . This framework is quite general; for example, observe that if $\boldsymbol{\Sigma}_i = \sigma^2 I$ for all i then maximizing (2.2) is equivalent to optimizing with the sum-of-squares criterion. Given this parameterization, a standard iterative *expectation-maximization* (EM) process can be used to find the values of $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, and γ .

The major problem with this model is that if the parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are allowed to vary freely for each distribution then finding a global optimum for (2.2) can be too time and space intensive for large problems. On the other hand, previously-suggested constraints, such as $\boldsymbol{\Sigma}_i = \sigma^2 I$, are too limiting. A major contribution of Banfield and Raftery's work is to reparameterize the distributions in a way which allows more flexibility in the characteristics of each distribution while still being solvable for larger problems than the unconstrained model. Specifically, they assume each distribution is multivariate normal as before, but decompose $\boldsymbol{\Sigma}_i$ via singular value decomposition as follows:

$$\boldsymbol{\Sigma}_i = D_i \Lambda_i D_i^T, \quad (2.3)$$

where D_i is the matrix of normalized eigenvectors and Λ_i is the diagonal matrix of eigenvalues of $\boldsymbol{\Sigma}_i$. Since the covariance matrix is positive semi-definite, this decomposition is always possible. Under this parameterization, the orientation of E_i is determined by D_i , and its shape and density contours are determined by Λ_i . The matrix Λ_i can then be further parameterized as $\lambda_i A_i$, where λ_i is the principal eigenvalue of $\boldsymbol{\Sigma}_i$. Hence λ_i determines the "volume" of each cluster and A_i determines the shape.

Banfield and Raftery use this parameterization to derive several constrained clustering models. They propose one model in which $A_i = I$ for all A_i ; under this assumption, the distributions are all spherical but have different sizes. Another proposed model is to constrain all the A_i 's to be equal, but allow λ_i and D_i to vary for each distribution. This results in clusters which have the same shape but different orientations and sizes. Finally, Banfield and Raftery demonstrate the derivation of a model which is not purely Gaussian, but instead has one designated dimension in which the points vary uniformly in a cluster-determined interval. While these constrained models do not apply to all sets of input, when

used appropriately they can significantly speed up the clustering process by reducing the number of model parameters.

Banfield and Raftery also derive a statistic called the *approximate weight of evidence* (AWE) which estimates the Bayesian posterior probability of a clustering solution produced via the above process. This statistic can be used to compare the results obtained with different mixture models and/or values of k . In subsequent work by Raftery and Fraley [13], the AWE statistic has been replaced by an alternative approximation called the *Bayesian information criterion* (BIC) derived by Schwarz [30].

Finally, a common criticism of mixture models (and related methods such as k -means) is that they do not address the problem of “noise”, defined loosely as input elements which do not belong to any cluster. Banfield and Raftery address this shortcoming by formally defining a notion of noise as a Poisson process with intensity ν which distributes points throughout the space containing all of the input points. Suppose that $\gamma_r = 0$ if \mathbf{x}_r is considered noise, and that E_0 is the set of all noise points. Then (2.2) can be generalized to

$$L(\theta, \nu, \gamma) = \frac{(\nu V)^{|E_0|} e^{-\nu V}}{|E_0|!} \prod_{r=1}^n f_{\gamma_r}(\mathbf{x}_r | \theta), \quad (2.4)$$

where V denotes the hypervolume of the space containing the input points.

Despite Banfield and Raftery’s attempts to make clustering via mixture models more practical, the technique is still relatively unknown compared to older approaches such as k -means and single link clustering. There are several probable causes.

First, the technique does not focus on efficiency. No attempt is made to analyze the time or space required for the algorithm as a function of input size, and the examples upon which the algorithm is run are in general quite small. Hence, clustering by mixture models may not be suitable for the currently trendy applications of clustering in computer science, since these often impose severe time and space limitations. At a minimum, further study should be conducted to determine the practical limits of the algorithm as a function of input size and dimensionality.

Second, a large degree of manual intervention is required. As currently implemented, the user is required to specify to the algorithm a model and a number of clusters along with the data set. The user can then vary the model and number of clusters and use the AWE/BIC statistic to compare the results. The user can also alter the results by selecting which attributes of the data to use as input, as well as how those attributes are represented. It would be interesting to see an attempt to make the computer optimize over **all** of these parameters simultaneously instead of splitting the task between the user and the computer. This idea will be discussed further in Section 3.2.

Finally, mixture models rely on the assumption that the data fits a Gaussian distribution. This may not be true in many cases; but even worse, the data may not be numerical, making

1. For each node v update w_v as follows:
 - For each tuple $\tau = \{v, u_1, \dots, u_{d-1}\}$ containing v do
 - $x_\tau \leftarrow \oplus(w_{u_1}, \dots, w_{u_{d-1}})$
 - $w_v \leftarrow \sum_\tau x_\tau$
2. Normalize the configuration such that the sum of the squares of the weights in each field is 1.

Figure 2: The configuration update procedure of Gibson *et. al.*. The combining operator \oplus can take on a number of different definitions.

this approach totally inapplicable. In particular, most databases contain a large amount of *categorical data*, meaning that each item has attributes which are drawn from a small set of possibilities over which there is no inherent notion of distance. Examples would be the color of a person’s hair or the manufacturer of a car. The next section describes an approach which specializes in this type of data.

2.2 Gibson, Kleinberg, and Raghavan: Dynamical Systems

This section describes the paper “Clustering Categorical Data: An Approach Based on Dynamical Systems” by Gibson, Kleinberg, and Raghavan [16]. Their approach relies on a simple process which iteratively computes weights on the vertices of a graph until a fixed point is reached; in this respect it is similar to the work of Kleinberg on finding authoritative web sources [22] and to the work of Teng *et. al.* on spectral graph partitioning [33, 26].

In this approach, the input data is a set $T = \{\tau_1, \tau_2, \dots, \tau_n\}$ of tuples with d fields, each of which can take one of a small set of values. A key assumption of this approach is that this set is not metric, so one can determine if two tuples have the same value for a particular field, but no additional information such as degree of similarity is associated with non-matching values.

A convenient way to view the input is as a graph in which the vertices are the set of all values appearing in the input tuples. In this view of the data, the values are referred to as *nodes* and the set of nodes is denoted $V = \{v_1, v_2, \dots, v_m\}$. Each tuple can be considered a path through the graph, and the edge set of the graph is simply the collection of edges in the paths. Associated with each node v in the graph is a weight w_v , and the vector \mathbf{w} containing the weights of all of the nodes is referred to as a *configuration* of the graph.

At the core of this approach is a function f which maps the current configuration to the next configuration. The idea is to repeatedly apply f until a fixed point (referred to by the authors as a *basin*) is reached; i.e., $f(\mathbf{w}) \approx \mathbf{w}$. The function f is defined procedurally

in Figure 2. In order to fully specify f , the *combining operator* \oplus must be defined. The operators considered in the paper include the following:

- The product operator \prod : $\oplus(w_1, \dots, w_d) = w_1 w_2 \cdots w_d$.
- The S_p operator: $\oplus(w_1, \dots, w_d) = ((w_1)^p + (w_2)^p + \cdots + (w_d)^p)^{1/p}$.
- The S_∞ operator: $\oplus(w_1, \dots, w_d) = \max\{|w_1|, \dots, |w_d|\}$.

In addition to choosing the combining operator, the user must also choose an initial configuration. The authors suggest two methods. In order to explore the data without introducing any bias, one can use random or uniform node weights for the initial configuration. On the other hand, to bias the results towards a grouping based on a certain set of values, one can give those values high initial weight.

Gibson *et. al.* claim that relatively little has been formally proved about the behavior of dynamical systems such as the one outlined above. However, they do present a few formal results in their paper. To avoid unnecessary repetition of the paper these results are summarized in a non-rigorous fashion below; see the paper for a more technical presentation.

Result 2.2.1 *For every set T of tuples and every initial configuration \mathbf{w} , $f^i(\mathbf{w})$ converges to a fixed point as $i \rightarrow \infty$ when S_1 is used as the combining rule.*

Result 2.2.2 *If T_a and T_b are two sets of tuples with no field values in common and the input set is $T_a \cup T_b$, the chance that the iterative process will converge to weights which “distinguish” the values in T_a from those in T_b grows quickly as the sets get more unequal in size.*

Result 2.2.3 *If the dynamical system has multiple basins, they can all be discovered by starting iteration with a “relatively small” set of random initial configurations.*

These results hint at a methodology for using this type of dynamical system for data exploration. One can apply the iterative process to a set of tuples until it converges (which is proven to happen in some cases by Result 2.2.1 above, and happens anecdotally for a wider range of combining rules according to the authors). At that point, some nodes will have high weights and some will have lower weights. Result 2.2.2 indicates that by separating tuples containing high weight nodes from those containing low weight nodes, one can often find a partition in the input set that has some semantic meaning. Again, the authors provide additional anecdotal evidence to support this conclusion. Finally, Result 2.2.3 shows that runs from different initial configurations can often find different meaningful groupings within the data. Since the theorems in the paper are not sufficient to fully support the behavioral claims, the authors present a large number of additional empirical results in support of this methodology.

In addition to this methodology, the authors suggest another approach inspired by spectral graph partitioning. Instead of operating on one configuration at a time, the user can maintain several configurations $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \dots, \mathbf{w}^{(m)}$ simultaneously. The user can then perform the following two steps until $\mathbf{w}^{(1)}$ achieves a fixed point:

1. Update $\mathbf{w}^{(i)} \leftarrow f(\mathbf{w}^{(i)})$ for $i = 1, 2, \dots, m$.
2. Update the set of vectors $\{\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \dots, \mathbf{w}^{(m)}\}$ to be orthonormal.

The second step in which the vectors are made orthonormal can introduce negative weights into the configurations; the authors claim that separating the positive weight nodes from the negative weight nodes in the various configurations can result in an informative partition of the data. They term configuration $\mathbf{w}^{(1)}$ after iteration the “principal basin”, and the others “non-principal basins.”

This approach shows several promising features. It converges quickly; generally less than 20 iterations are required in most of the examples in the paper. The authors also demonstrate that it can identify “clusters” even in the presence of irrelevant fields and large numbers of randomly generated “noise” inputs. Also, the non-principal basins can apparently be used to separate multiple clusters within the same input set.

Nevertheless, this approach has several weaknesses. The most basic is the current lack of both theoretical and practical understanding of the behavior of the algorithm. Theoretically, convergence is not guaranteed with most combining operators, and there are few results characterizing exactly what types of clusters may be identified by this method. Partially because of the lack of theoretical understanding, there is no clearly defined methodology to use this approach on practical clustering algorithms. The authors allude to “masking” and “augmenting” certain terms in the combining operator, but do not provide insight into when or how to perform these modifications. They also provide little in the way of insight on how to choose the right combining operator or initial state for a specific task. Until these issues are addressed, this approach is likely to be little more than a curiosity item.

Additionally, the fact that the approach deals only with categorical data can be a drawback. Categorical data is more general than numerical data, because even numerical vectors can be treated as categorical data. However, doing so discards useful similarity/distance information, and should probably be avoided if possible. The next section describes a graph-based approach, which unlike the papers considered so far, can deal with any type of data for which there is a well-defined notion of similarity.

2.3 Ben-Dor and Yakhini: Clique Graphs

Clustering has recently become popular in computational biology as a technique for analyzing DNA microarray data [8, 3, 19]. A pair of recent papers, one by Ben-Dor and

Yakhini [3] the other by Hartuv *et. al.* [19], stand out in this literature because they describe novel clustering algorithms that could potentially be useful for many applications. The papers contain many similar ideas, but this section will focus exclusively on the work by Ben-Dor and Yakhini.

Underlying Ben-Dor and Yakhini’s approach is a model of input called the *corrupted clique model*. The basic idea of this model is that ideally, all elements within each cluster should be similar to one another, and not similar to any elements of other clusters. Hence, the similarity graph for the data should appear as a set of vertex-disjoint cliques; this type of graph is called a *clique graph*. In practice, however, the similarity relation is usually approximate, so the actual similarity graph derived from the data has some extra and some missing edges in it when compared to the ideal similarity graph. In the corrupted clique model, one assumes that the input is a clique graph that has been corrupted by adding each non-existing edge with probability α and removing each existing edge with probability α . The goal of the clustering algorithm is thus to find the original clique graph given the corrupted version.

Ben-Dor and Yakhini present two algorithms to accomplish this task. The first is a theoretical algorithm, about which they prove several performance results. The second is a practical heuristic implementation based on the ideas of the theoretical algorithm.

Several definitions are required to describe the theoretical algorithm.¹ Suppose that V denotes the set of vertices in the input graph, and V is a union of disjoint sets so $V = E_1 \cup E_2 \cup \dots \cup E_k$ where each E_i represents a cluster. Suppose that for $v \in V$, $C(v) = i$ if and only if $v \in E_i$, so $C(v)$ is the label of the cluster which contains v .

A *core* V' of V is defined as $V' = E'_1 \cup E'_2 \cup \dots \cup E'_k$, where each E'_i is a non-empty subset of E_i . The core V' *classifies* an input $v \in V - V'$ with the following function:

$$C_{V'}(v) = \max_{1 \leq i \leq k} \deg(v, E'_i) / |E'_i|. \quad (2.5)$$

Intuitively, $C_{V'}(v)$ represents the cluster which appears to contain v based only on the similarity of v to elements in the core. Hence, if $C_{V'}(v) = C(v)$, then V' correctly classifies v . The essence of the clustering problem in this approach is to identify a small core which correctly classifies all of the input elements.

The theoretical algorithm relies upon the following result, stated non-technically here for brevity.

Result 2.3.1 *Let m denote the size of the smallest cluster in the input. If α is not too large and m is not too small, then with high probability a random small subset of V contains a core which correctly classifies all inputs.*

Based on this result, the theoretical algorithm can be summarized as follows:

¹In order to be more consistent with the other sections, the notation in this section has been changed significantly from that in the original paper.

1. Consider a small subset V' of V .
2. Consider all ways of partitioning V' into k non-empty clusters. Each such partition is called a *core candidate*.
3. With each core candidate, classify all of the remaining points. Keep the classification which results in the clique graph most similar to the input graph.

The practical algorithm, which the authors call CAST, works slightly differently. It requires no prior knowledge of the number of clusters, and avoids brute-force enumeration of core candidates. It takes as input a similarity function $s : V \times V \rightarrow [0, 1]$ and a coefficient t and defines the *affinity* $a(v, E)$ of an element v to a cluster E as

$$a(v, E) = \sum_{u \in E} s(u, v). \quad (2.6)$$

An element v is said to have high affinity for E if $a(v, E) \geq t|E|$; otherwise it has *low affinity* for E . The algorithm operates by constructing one cluster at a time. In general, it alternates between adding high affinity elements to a cluster and removing low affinity elements. When this process stabilizes, the cluster is considered finished, and a new cluster is started. The process stops when all elements have been assigned to a cluster.

The authors present results of the practical algorithm on both simulated and real biological data; the results on simulated data are impressive in terms of accuracy, but do not report the running time or storage requirements of the algorithm. Hence, the scalability of the method is difficult to determine. However, the concept of identifying a high-quality “core” with a small number of elements which can be used to classify the remaining points hints at a useful way of scaling the algorithm: use a relatively small set of elements to generate the core, then use a simpler strategy to classify the remaining inputs and adjust the clusters as needed. A similar strategy has been successfully applied in the data mining community in systems such as CLARANS [28] and BIRCH [39].

Otherwise, the algorithm seems very appealing for data whose similarity patterns fit the corrupted clique model. However, it is important to note that by assuming a similarity matrix as input, the authors have avoided addressing issues such as how to select which attributes to use for clustering and how to represent them to the algorithm. The next section describes another general clustering algorithm that addresses these and other pragmatic concerns.

2.4 Agrawal, Gehkre, et. al.: Subspace Clustering

This section describes the paper “Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications” by Agrawal, Gehrke, Gunopulos, and Raghavan [1]. The paper describes the CLIQUE clustering algorithm, which is targeted specifically at data

mining large databases. In particular, the authors identify three goals for practical data mining systems and argue that CLIQUE satisfies all of them. These goals are summarized below:

Effective Treatment of High Dimensionality: Each item in a database may have a large number of attributes, many of which may be irrelevant or misleading with respect to the formation of clusters.

Interpretability of Results: Many applications require that the algorithm produce a simple “summary” of each cluster for the user.

Scalability and Usability: The clustering algorithm must be fast and easy to use even on large databases, and be insensitive to noise and to the order in which the data is read.

The CLIQUE system takes a three step approach to clustering. First, a set of “subspaces” is chosen in which to cluster the data. Then, clustering is performed independently in each subspace. Finally, a compact summary of each cluster is generated in the form of a disjunctive normal form (DNF) expression.

Suppose the input is a set of n d -dimensional numerical² vectors from the space $\mathcal{A} = A_1 \times A_2 \times \dots \times A_d$, where each A_i is an interval. Suppose furthermore that each A_i is partitioned into ξ subintervals, where ξ is a user-supplied parameter. Applying such partitions in the original d -space divides the space into a set of d -dimensional axis-aligned rectangles which are termed *regions* by the authors. A region is termed *dense* if the proportion of input vectors contained within the region exceeds τ , another user-supplied parameter.

Now consider a graph in which the vertices are dense regions, and there is an edge between vertices if the corresponding regions in space share a face. A *cluster* can then be succinctly defined as a connected component in this graph. Also, observe that a region can be described simply as a conjunction of range checks in each of the dimensions; hence, a cluster can be described as a disjunction of the conjunctions describing the regions which comprise it.

These definitions are sufficient to describe how the authors perform clustering in the full-dimensional input space. First, they identify the dense regions, then use the above graph representation to find the clusters. Second, a DNF formula is generated for each cluster by taking a disjunction of the descriptions of each region in the cluster and then heuristically simplifying the resulting expression.

However, the authors are not content to perform clustering only on the full-dimensional data; they also want to consider what they call the *subspaces* of the original data. Specifically, the input space \mathcal{A} has 2^d subspaces which can be formed by selecting all possible

²The authors show that the method can be modified to handle categorical data, but presenting it in the context of numerical data is more intuitive.

subsets of the original dimensions to represent the input data. During a preprocessing step, the authors use an algorithm to identify every subspace which contains at least one dense region. The clustering stages defined above are then run separately on each such subspace. Because it may have to consider every possible subspace of the original dimensions, this preprocessing step takes exponential time in the number of dimensions. However, the authors argue that using various tricks and heuristics one can keep the actual time “reasonable” on real data.

Instead of relying on subspaces, traditional approaches to dimensional reduction have relied upon singular value decomposition of the original data [7, 14]. The idea is that the data can be transformed to the coordinate system defined by its eigenvectors, and then those axes corresponding to small eigenvalues can be discarded. The authors argue that this approach complicates the analysis of the final clusters, since they are formed in terms of linear combinations of the original input values. They also demonstrate several situations in which “obvious” clusters are obscured by the SVD approach.

The main contribution of this paper seems to lie in its concrete definitions of the desirable properties of approaches to data mining, and the unique concept of searching for clusters in the subspaces of the input set. However, the specific approach taken by CLIQUE is easily criticized. It makes use of an exponential time algorithm to analyze the subspace structure. In addition, the parameters ξ and τ seem obscure, difficult to choose, and potentially limiting. For example different clusters may have different “densities”, making a single value of τ for the whole data set impractical, and using the wrong value can obviously be disastrous. Similarly, differences in the distributions in each dimension may call for different optimal values of ξ . On the other hand, adding even more parameters such as a separate value of ξ for each dimension simply adds to the difficulty of tuning the algorithm for a data set.

3 Analysis and Conclusions

3.1 Choosing a Clustering Algorithm

Choosing a clustering algorithm for a particular problem can be a daunting task. Because clustering has been addressed in a wide variety of disciplines, it can be hard to even find the most relevant approaches to a particular problem. This difficulty is exacerbated by the fact that different disciplines use different terminology and basic definitions to describe their approaches.

Below is a list of criteria which is crucial to choosing a clustering algorithm. Included in the description of each criterion is an analysis of which clustering algorithms discussed in this paper best addresses the various situations which can arise under each criterion. The astute reader will notice that there are still tradeoffs involved in choosing a clustering approach; even analyzing a clustering problem with respect to the criteria below may not

yield a single “best” algorithm to choose. The details of the discussion below is summarized in Figure 3.

Data and Cluster Model: Perhaps the most important criterion for matching an instance of a clustering problem to a clustering algorithm is the nature of the data and the anticipated clusters. As previously noted, many clustering algorithms assume a certain type of input such as numerical input in the case of k -means or categorical input as in Gibson’s paper. Others require the availability of a distance metric or similarity measure for the data. Similarly, many algorithms implicitly assume that the cluster structure has certain characteristics. For example, in the case of k -means the clusters are assumed to be spherical, and in the case of single-link hierarchical clustering the clusters are assumed to be “well separated” so that incorrect links are not made.

Analysis based on these ideas can be applied to the algorithms in Section 2. If the data is numerical and the clusters are believed to be spherical or ellipsoidal, then approaches based on mixture models may be appropriate. On the other hand, if the data is non-numerical and/or little is known *a priori* about the geometry of the clusters, one of the graph-based approaches may be more appropriate.

In general, categorical data is best handled by the approaches of Ben-Dor and Gibson. The former is probably the best possibility when a well-defined notion of similarity is available for the data in question, while the latter is a better heuristic approach for “exploring” data and finding clusters which arise from the co-occurrence of certain categorical values.

Mixed data is handled best by the approaches of Ben-Dor and Agrawal. Ben-Dor’s more rigorous approach once again has the advantage in the presence of a known similarity measure for the data, whereas Agrawal’s approach is best used when it is unclear which attributes of the data should be examined while clustering.

The most general case, in which little is known about the data or the clusters, is not handled well by any of the algorithms since they all require that the user have sufficient knowledge to choose appropriate input parameters. This unfortunate case is discussed more fully in Section 3.2 below.

Scalability: Despite the ongoing exponential increases in the power of computers, scalability remains still a major issue in many clustering applications. In commercial data mining applications, the quantity of the data to be clustered can far exceed the main memory capacity of the computer, making both time and space efficiency critical; this issue is addressed by clustering systems in the database community such as BIRCH [39]. In other cases such as clustering algorithms imbedded within World

Wide Web search engines, time efficiency is imperative because search engine users want results very quickly.

Scalability is not directly addressed in Banfield or Ben-Dor’s papers, while the other two papers present empirical results showing that their algorithms can handle large input sets, although the time required by Agrawal’s approach grows quickly with the number of dimensions of the input. Statistical mixture models often require quadratic space and the EM algorithm converges relatively slowly, making scalability an issue. Ben-Dor’s theoretical algorithm makes use of sampling to reduce the amount of work needed to form the “cores” of each cluster, but the practical algorithm does not use of this methodology. Of course, sampling can be used to scale any clustering algorithm; this idea will also be mentioned again in Section 3.2.

Noise: While traditional methods such as those described in Section 1 tend to ignore the problem of noise, all of the algorithms presented in Section 2 deal with noise in some way or another. In the mixture model approach, noise is modeled as a Poisson process and the clustering procedure explicitly labels some inputs as noise. In the dynamical systems and subspace approaches, noise points are simply left out of the clusters. Specifically, in the former approach the authors claim that these elements do not achieve high enough weights to be placed in clusters, while in the latter noise elements tend to fall outside “dense regions” and are thus ignored. Finally, Ben-Dor’s system does not specifically address the issue of noise, but under the clique graph model, noise points will tend to fall into extremely small clusters (often singletons) which can subsequently be ignored.

Result Presentation: In many practical clustering applications, it is useful if a succinct “summary” of each cluster can be given to the user. All of the systems within this paper except Ben-Dor’s have this ability. Banfield’s method generates a centroid and covariance matrix for each cluster, Gibson’s dynamical system identifies a set of high-weight values around which each cluster is formed, and Agrawal’s approach produces a DNF expression summarizing each cluster. Perhaps a post-processing phase could be added to Ben-Dor’s algorithm to choose a representative “median element” for each cluster.

3.2 Future Directions for Clustering Research

As the previous section illustrates, one of the major challenges in using a clustering algorithm on a specific problem lies not in performing the clustering itself, but rather in choosing the algorithm and the values of the associated parameters. Each of the papers discussed in Section 2 contain enough positive empirical results that it is easy to conclude that current

Approach	Data	Scales	Noise	Parameters
Mixture models	Numerical	Poorly	Models noise as a Poisson process and explicitly identifies inputs as noise.	X The input set with dimensions and units selected appropriately <hr/> k The number of clusters <hr/> θ The model to be used and its corresponding parameters
Dynamical systems	Categorical	Well	The nature of the dynamical system prevents noisy inputs from gaining high weight	T the input set <hr/> \mathbf{w} The initial config (or set of initial configs) <hr/> \oplus The combining operator
Clique graphs	General	Unknown	Noise points end up in small clusters	S A similarity matrix <hr/> t The “affinity coefficient”
Subspace clustering	General	Well with number of inputs but very poorly with number of dimensions	Noise points are those which lie outside “dense” regions	T The input set with units selected appropriately <hr/> ξ The number of partitions per dimension <hr/> τ The “density cutoff”

Figure 3: Summary of the relevant features of the clustering algorithms from Section 2.

clustering algorithms do a good job **when run on appropriate data with the appropriate parameters**. It is choosing the algorithm and the parameters that present the challenge; in particular, a quick scan of the “Parameters” column in Figure 2 demonstrates that current approaches often rely on non-intuitive parameters which may be difficult for even the informed user to select.

This leads to the conclusion that there are three general categories of clustering algorithms. First, there is **classification in the presence of prior knowledge** about the clusters; this is often called “supervised learning” and is not examined directly in this paper. Second, there is **classification to test a hypothesis**. In this instance, the user has formed a hypothesis that under a certain set of assumptions about the data, the data can easily be organized into distinct clusters. This hypothesis is the basis for selecting an appropriate clustering algorithm, data representation, and set of algorithm parameters. If the hypothesis is valid then the algorithm should produce “good clusters”; otherwise the hypothesis requires modification. It is at this type of clustering that the methods in this paper excel.

Finally, there is **classification to explore a data set**, which hereafter will be referred to as *data mining*. In this case the user has no *a priori* assumptions about the data, but wants to know if it falls into “meaningful groups” (a term for which the user may not even have a specific definition). Data mining is currently a trendy application of clustering technology, but very little work has been done to gracefully eliminate fixed input parameters to clustering algorithms. Thus, most data mining approaches implicitly require the user to alternate between running the clustering algorithm, modifying the parameters, and choosing the results which seem “best.”

This leads to the following set of proposals for continuing research in clustering, and in particular data mining:

- *Gracefully eliminate the need for a priori assumptions about the data.* An obvious candidate for elimination is fixed numerical inputs (such as ξ in Agrawal’s subspace clustering approach). Obviously, one can define a notion of output quality, as Banfield and Raftery have done with their AWE statistic, and then iterate over many possible parameter values searching for a maximum. This trivially can eliminate the need for fixed inputs, at the cost of a potentially large increase in running time. More intelligent strategies than brute-force sampling to choose parameters would be helpful.

Additionally, current clustering algorithms make implicit assumptions about the input data. Common examples of such assumptions are that the units in each dimension are scaled appropriately, that the predefined notion of distance/similarity is ideal, and/or that the set of data attributes used for clustering are correct (although the approaches by Gibson and Agrawal are notable for attempting to avoid this assumption). An interesting proposal would be to explore useful ways to explicitly parameterize these assumptions and allow the computer to search for good specific choices. For example,

the computer could be allowed to select which dimensions of the data to use in the similarity calculation, or to choose whether or not to normalize the data in some dimension.

- *Use sampling to improve efficiency.* Theoretically-founded clustering algorithms are often dismissed as too inefficient to be used in practice. This problem will be exacerbated if the previous proposal is followed and the computer is forced to search over an even broader array of choices in the pursuit of “good clustering.” The theoretical results of Ben-Dor show that with a good choice of a similarity measure, forming a set of “cluster cores” and then classifying the remaining inputs is a very effective strategy as long as the sample size is big enough to get representatives from every important cluster. This strategy could be pursued to help “scale up” computationally intensive clustering methods.

3.3 Conclusion

This paper has described four recently-developed approaches to clustering. Each has both positive and negative aspects, and each is suitable for different types of data and different assumptions about the cluster structure of the input.

However, all of the algorithms still rely to some extent on rigid assumptions about the data which must be provided by the user. These assumptions appear both explicitly as fixed numerical parameters and implicitly in the way which the user represents the input to the algorithm. While prior knowledge of the application can help the user choose appropriate assumptions, oftentimes the user must still alternate between running the clustering algorithm and updating the assumptions based on the result.

The paper concludes by suggesting a new general strategy for clustering as a method of data mining. In this strategy, the assumptions required by the various clustering algorithms can be explicitly parameterized, allowing the computer more freedom to search for the best way to cluster the data. To compensate for the broadening of search space of possible clusterings, it is recommended that an implementation of this strategy use sampling to reduce the number of inputs if necessary. It is hoped that future work will lead to a specific clustering algorithm based on these ideas that can then be validated on actual data.

References

- [1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD 98)*, pages 94–104, 1998.

- [2] Jeffrey D. Banfield and Adrian E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49:803–821, September 1993.
- [3] Amir Ben-Dor and Zohar Yakhini. Clustering gene expression patterns. In *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB 99)*, 1999.
- [4] Gilles Celeux and Gérard Govaert. Gaussian parsimonious clustering models. *Pattern Recognition*, 28(5):781–793, 1995.
- [5] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *15th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.
- [6] D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20:364–366, 1977.
- [7] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.
- [8] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95:14863–14868, December 1998.
- [9] A. El-Hamdouchi and Peter Willet. Hierarchic document clustering using ward’s method. In *Proceedings of the Ninth International Conference on Research and Development in Information Retrieval*, pages 149–156, 1986.
- [10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD 96)*, 1996.
- [11] Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. In *Proceedings of the Fourth International Symposium on Large Spatial Databases (SDD 95)*, 1995.
- [12] Brian S. Everitt. *Cluster Analysis*. Halsted Press, third edition, 1993.
- [13] C. Fraley and A. E. Raftery. How many clusters? which cluster method? answers via model-based cluster analysis. Technical Report 329, Department of Statistics, University of Washington, February 1998.

- [14] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.
- [15] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
- [16] D. Gibson, J. Kleinberg, and P. Raghavan. Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings of the 29th ACM Symposium on Theory of Computing (STOC 97)*, 1997.
- [17] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.
- [18] A. D. Gordon. *Classification: Methods for the Exploratory Analysis of Multivariate Data*. Chapman and Hall, 1981.
- [19] Erez Hartuv, Armin Schmitt, Jörg Lang, et al. An algorithm for clustering cdnas for gene expression analysis. In *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB 99)*, 1999.
- [20] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
- [21] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Inc., 1990.
- [22] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 1998.
- [23] Jon Kleinberg, Christos Papadimitriou, and Prabhakar Raghavan. Segmentation problems. In *Proceedings of the 30th ACM Symposium on Theory of Computing (STOC 98)*, pages 473–481, 1998.
- [24] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proc. 5th Berkeley Symposium*, 1:281–297, 1967.
- [25] David W. Matula. *Classification and Clustering*, chapter Graph Theoretic Techniques for Cluster Analysis Algorithms. Academic Press, Inc., 1977.
- [26] Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *Journal of the ACM*, 44(1):1–29, January 1997.
- [27] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26:354–359, 1983.

- [28] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB 94)*, 1994.
- [29] Edie Rasmussen. *Information Retrieval*, chapter Clustering Algorithms, pages 419–442. Prentice Hall, 1992.
- [30] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.
- [31] A. J. Scott and M. J. Symons. Clustering methods based on likelihood ratio criteria. *Biometrics*, 27:387–397, 1971.
- [32] R. Sibson. Slink: an optimally efficient algorithm for a complete link method. *The Computer Journal*, 16:30–34, 1973.
- [33] Daniel A. Spielman and Shang-Hua Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 1996.
- [34] M. J. Symons. Clustering criteria and multivariate normal mixtures. *Biometrics*, 37:35–43, 1981.
- [35] Ellen M. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing & Management*, 22(6):465–476, 1986.
- [36] J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):235–244, 1963.
- [37] Peter Willet. Recent trends in hierarchical document clustering: A critical review. *Information Processing & Management*, 24(5):577–597, 1988.
- [38] Zhigang Xiang. Color image quantization by minimizing the maximum intercluster distance. *ACM Transactions on Graphics*, 16(3):260–276, July 1997.
- [39] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD 96)*, pages 103–114, 1996.