The Dissertation Committee for Alexander Strehl certifies that this is the approved version of the following dissertation:

Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining

Committee:

Joydeep Ghosh

Anthony P. Ambler

Ross Baldick

Inderjit S. Dhillon

Raymond J. Mooney

S. Lynne Stokes

Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining

by

Alexander Strehl, 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

May 2002

Dedicated to my wonderful parents
for their love, patience and measureless support

Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining

Publication No. _____

Alexander Strehl, Ph.D.
The University of Texas at Austin, 2002

Supervisor: Joydeep Ghosh

This dissertation takes a *relationship-based approach* to cluster analysis of high (1000 and more) dimensional data that side-steps the 'curse of dimensionality' issue by working in a suitable similarity space instead of the original feature space. We propose two frameworks that leverage graph algorithms to achieve relationship-based clustering and visualization, respectively. In the visualization framework, the output from the clustering algorithm is used to reorder the data points so that the resulting permuted similarity matrix can be readily visualized in 2 dimensions, with clusters showing up as bands. Results on retail transaction, document (bag-of-words), and web-log data show that our approach can yield superior results while also taking additional balance constraints into account.

The choice of similarity is a critical step in relationship-based clustering and this motivates our systematic comparative study of the *impact of similarity* measures on the *quality of document clusters*. The key findings of our experimental study are: (i) Cosine, correlation, and extended Jaccard similarities perform comparably; (ii) Euclidean distances do not work well; (iii) graph partitioning tends to be superior to k-means and SOMs especially when balanced clusters are desired; and (iv) performance curves generally do *not* cross. We also propose a cluster quality evaluation measure based on normalized mutual information and find an analytical relation between similarity measures.

It is widely recognized that combining multiple classification or regression models typically provides superior results compared to using a single, well-tuned model. However, there are no well known approaches to combining multiple clusterings. The idea of combining cluster labelings without accessing the original features leads to a general knowledge reuse framework that we call *cluster ensembles*. We propose a formal definition of the cluster ensemble as an optimization problem. Taking a relationship-based approach we propose three effective and efficient combining algorithms for solving it heuristically based on a hypergraph model. Results on synthetic as well as real data-sets show that cluster ensembles can (i) improve quality and robustness, and (ii) enable distributed clustering, and (iii) speed up processing significantly with little loss in quality.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

> This morning I declined to write a popular article about the question "Can machines think?" I told the editor that I thought the question as ill-posed and uninteresting as the question "Can submarines swim?"
>
> – Edsger W. Dijkstra[1]

1.1 Cluster Analysis

The ability to form meaningful groups of objects is one of the most fundamental modes of intelligence. Humans perform this task with remarkable ease. In early childhood one learns to distinguish, for example, between cats and dogs or apples and oranges. However, enabling the computer to do this task of grouping automatically is a difficult and often ill-posed problem.

Cluster analysis is a tool for exploring the structure of data. The core of cluster analysis is *clustering*, the process of grouping objects into *clusters*

[1] In *The Fruits of Misunderstanding*, EWD 854, May 1983

such that objects from the same cluster are similar and objects from different clusters are dissimilar. Objects can be described in terms of measurements (e.g., attributes, features) or by relationships with other objects (e.g., pairwise distance, similarity). Unlike classification, clustering does not require assumptions about category labels that tag objects with prior identifiers. Therefore, clustering is an *unsupervised learning* technique versus classification, which belongs to supervised learning.

The need to structure and learn from the vigorously growing amounts of data has been a driving force for making clustering a highly active research area. Humans are not able to easily discover knowledge from the glut of information in databases without the use of summarization techniques. Basic statistics (such as mean, variance) or histograms can provide an initial feel for the data. However, more intricate relationships among the objects, among the features, and between both can be discovered through cluster analysis.

Clustering is used in many areas, including artificial intelligence, biology, customer relationship management, data compression, data mining, information retrieval, image processing, machine learning, marketing, medicine, pattern recognition, psychology, recommender systems and statistics. In biology, clustering is used, for example, to automatically build a taxonomy of species based on their features. Currently, there is considerable interest in estimation of phylogenetic trees from gene sequence data. Another application of clustering is to better understand gene functions in the biological processes in a cell. A key step in the analysis of gene expression data is the detection of groups of genes that manifest similar expression patterns. Another growing application area is customer relationship management, where data collected from multiple touch-points (e.g., web surfing, cash register transactions, call

center activities) has become readily available. This data contains valuable knowledge of customer behavior that can help optimize marketing, bundling, and pricing strategies. Due to the massive amounts and great detail of data, extracting such knowledge is hard, and often even obvious insights are overlooked. Clustering is critical in the mining process because it can summarize data to a manageable level by forming, for example, groups of customers with similar profiles.

Approaches to clustering differ in a variety of aspects. Clustering algorithms can proceed *divisively* (top-down) or *agglomeratively* (bottom-up). In top-down processing, one starts with the objects as one group and splits it into clusters. Conversely, in agglomerative algorithms, each object starts as a singleton cluster and clusters are merged successively. Depending on the result of clustering, one can distinguish between *hierarchical* (multi-layer) and *flat* (single-layer) techniques. Hierarchical algorithms output a rooted tree structure that can be represented as a dendrogram. Within hierarchical clustering, techniques can be distinguished into those generating binary trees and those generating arbitrary trees. In the case of flat clustering, *hard* clustering induces a partitioning into non-overlapping groups. In contrast to this Boolean membership relation, a *soft* clustering gives the probabilities for each object being a member of a cluster. Based on the goals of clustering, it can be formalized as an optimization problem by itself (e.g., minimizing squared error) or in the context of an application (facilitating for further activities such as predictive modeling or browsing).

Clustering has been widely studied in several disciplines, especially since the early 1960's. A variety of classic [Fuk72, Eve74, Har75, Mur85, JD88] and recent books [KR90, HK00, DHS01] give great introductions to approaches

and algorithms in cluster analysis.

The rest of this chapter is organized as follows. The next section introduces some basic notational conventions for this dissertation. Section 1.3 introduces our basic approach to clustering and the components involved. Section 1.4 elaborates on the motivations for this dissertation. In section 1.5, a brief summarization of the contributions is given. Section 1.6 contains the organization of the rest of this dissertation.

1.2 Notation

As the context and convention permits, small Roman and Greek letters are used for scalars, small Roman boldface letters for vectors, capital Roman boldface for matrices, and capital Greek for functions. Letters early in the alphabet tend to indicate constants and letters towards the end indicate variables. Capital caligraphs indicate sets and $|\bullet|$ their cardinality. $\|\bullet\|$ is the L_2 norm and bracketed super-scripts are indices rather than the power function. The letters K, N, D are in general scalars and refer to particular versions of k, n, d, respectively. $\mathbf{I}$ denotes the identity matrix, $\mathbf{0}$ and $\mathbf{1}$ are the vector / matrix equivalents to 0 and 1. $O(\bullet)$ is the Landau symbol. For a $d \times n$ matrix $\mathbf{X}$, $x_{i,j}$ refers to the element at the i-th row and j-th column and $\mathbf{x}_j$ ($\mathbf{x}_{\dagger i}$) is the j-th $d \times 1$ *column* ($n \times 1$ *row*) vector of $\mathbf{X}$. $\mathbf{X}^\dagger$ denotes the transpose of $\mathbf{X}$. The set of real numbers is $\mathbb{R}$, the set of non-negative (strictly positive) real numbers is $\mathbb{R}_+$ ($\mathbb{R}_{++}$).

1.3 Relationship-based Clustering Approach

The pattern recognition process from raw data to knowledge is characterized by a large reduction of description length by multiple orders of magnitude. A step-wise approach that includes several distinct levels of abstraction is generally adopted. A common framework can be observed for most systems as the input is transformed through the following spaces[2]:

Input space $\mathcal{I}$. Clustering is based on observations about the objects under consideration. The input space captures all the known raw information about the object, such as a customer's purchase history, a hypertext document, a grid of gray-level values for a visual image, or a sequence of DNA. Let N denote the number of objects in the input space, and let a particular object have D associated attributes. The number of attributes can vary by object. For example, objects can be characterized by variable length sequences.

Feature space $\mathcal{F}$. The transformation $\Upsilon : \mathcal{I} \rightarrow \mathcal{F}$ removes some redundancy by feature *selection* or *extraction*. In feature selection, a subset of the input dimensions that is highly relevant to the learning problem is selected. In document clustering, for example, a variety of word selection and weighting schemes have been proposed [YP97, Kol97]. A feature extraction transformation may extract the locations of edge elements from a gray-level image. Other popular transformations include the Fourier transformation for speech processing and the Singular Value Decomposition (SVD) for multivariate statistical data. Features (or attributes)

[2] A similar overview on spaces in classification problems can be found e.g. in [Kum00]

may be distinguished by the type and number of values they can take: We distinguish binary (Boolean true / false decision, such as `married`), nominal (finite number of categorical labels with no inherent order, such as `color`), ordinal (mappable to the cardinal numbers with a smallest element and an ordering relation, such as `rank`), continuous (real valued measurements such as `latitude`), and combinations thereof. The transformation to feature space may render some objects unusable, due to missing data or non-compliance with input constraints. Thus, we denote the corresponding number of objects as n ($n \leq N$) and the number of dimensions by d ($d \leq D$).

Similarity space $\mathcal{S}$ is an optional intermediate space between the feature space $\mathcal{F}$ and the output space $\mathcal{O}$. The similarity transformation $\Psi : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{S}$ translates a pair of internal, *object-centered* descriptions in terms of features into an external, *relationship-oriented* space $\mathcal{S}$. While there are n d-dimensional descriptions (e.g., $d \times n$ matrix $\mathbf{X} \in \mathcal{F}^n$) there are $(n-1)n/2$ pairwise relations (e.g., symmetric $n \times n$ matrix $\mathbf{S} \in \mathcal{S}^{n \times n}$). In our work, we mainly use similarities $s(\mathbf{x}_a, \mathbf{x}_b)$ because they induce mathematically convenient and efficient sparse matrix constructs. Instead of minimizing the cost, we maximize accumulated similarity. The dual notions of distance and similarity can be interchanged. However, their conversion has to be done carefully in order to preserve critical properties as will be discussed later. Similarities $s \in [0, 1] \subset \mathbb{R}$ and distances $d \in [0, \infty) \subset \mathbb{R}$ can be related in various non-linear, monotonically decreasing ways.

Output space $\mathcal{O}$. In partitioning, the output space is a vector of nominal

attributes providing cluster membership to one of k clusters. To represent a clustering, we denote it either as a set of clusters $\{\mathcal{C}_\ell\}_{\ell=1}^{k}$ or as a n-dimensional label vector λ, where $\mathbf{x}_j \in \mathcal{C}_\ell \Leftrightarrow \lambda_j = \ell$. Many approaches to the output transformation $\Phi : \mathcal{F}^n \to \mathcal{O}^n$ (or $\Phi : \mathcal{S}^{n\times n} \to \mathcal{O}^n$), the actual clustering algorithm, have been investigated. The next chapter will give an overview.

Hypothesis space $\mathcal{H}$ is the space where a particular clustering algorithm searches for a solution. $\mathcal{H}$ depends on the *language bias* of the clustering algorithm. A given algorithm can be viewed as looking for an optimal solution (at maximum objective or minimum cost as given by the evaluation function) according to its *search bias*. An evaluation function $\phi : \mathcal{H} \to \mathbb{R}_+$ can work purely based on the feature and/or similarity space, but can also incorporate external knowledge (such as user given categorizations).

Let us look at the output space in a little more detail. In this work, we focus on flat clusterings (partitions) for a variety of reasons. Any hierarchical clustering can be conducted as a series of flat clusterings, rendering flat clustering the more fundamental step. Hierarchical complete v-ary trees of depth τ can be encoded without loss of information as a single ordered flat labeling with $k = v^\tau$ clusters. To obtain the clustering on layer $\rho \in \{0, \ldots, \tau\}$, each of the v^ρ disjoint intervals of labels (with length $v^{\tau-\rho}$ starting at label 1) have to be collapsed to a single label (considered one cluster). (consider e.g., $v = 2$, $\tau = 3$, so the labels are structured in a tree as follows: $\{\{\{1, 2\}, \{3, 4\}\}, \{\{5, 6\}, \{7, 8\}\}\}$) Consequently, a flat labeling with implicit ordering information is just as expressive as full hierarchical cluster trees. But let us return to labelings without

ordering information.

Figure 1.1 illustrates all clusterings for less than 6 objects and clusters. For example, there are 90 partitionings of 6 objects into 3 groups. In general, for a flat clustering of n objects into k partitions there are

$$\frac{1}{k!}\sum_{\ell=1}^{k}\left(\begin{array}{c} k \\ \ell \end{array}\right)(-1)^{k-\ell}\ell^n \tag{1.1}$$

possible partitionings, or approximately $k^n/k!$ for $n \gg k$. Clearly, the exponentially growing search space makes an exhaustive, global search prohibitive. In general, clustering problems have difficult, non-convex, objective functions modeling the similarity within clusters and the dissimilarity between clusters. In general, the clustering problem is NP-hard [HJ97].

In this dissertation, the focus is on the similarity space. Most standard algorithms spend little attention on the similarity space. Rather, similarity computations are directly integrated into the clustering algorithms which proceeds straight from feature space to the output space. The introduction of an independent modular similarity space has many advantages, as it allows us to address many of the challenges discussed in the next section. We call this approach similarity-based or relationship-based or graph-based. Figure 1.2 contrasts the object-centered with the relationship-based approach. The key difference between relationship-based clustering and regular clustering is the focus on the similarity space $\mathcal{S}$ instead of working directly in the feature domain $\mathcal{F}$. In figure 1.3 an abstract overview of the general relationship-based framework is shown. In web-page clustering, for example, $\mathcal{X}$ is a collection of n web-pages. Extracting features yields $\mathbf{X}$, for example, the term frequencies of stemmed words, normalized such that $\forall \mathbf{x} \|\mathbf{x}\|_2 = 1$. Similarities are computed, using e.g., cosine based similarity Ψ yielding the $n \times n$ similarity matrix $\mathbf{S}$.

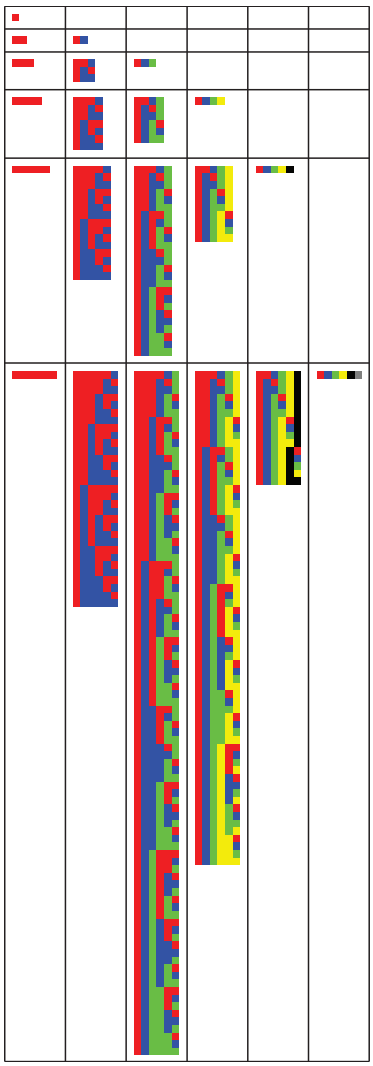


Figure 1.1: All possible clusterings of up to $n = 6$ objects (rows top to bottom) into up to $k = 6$ groups (columns left to right). In each table cell a matrix shows all clusterings for a particular choice of n and k. A matrix shows one clustering per row. The color in the j-th column indicates the group membership of the j-th object. Group association is shown in red, blue, green, black, and gray.

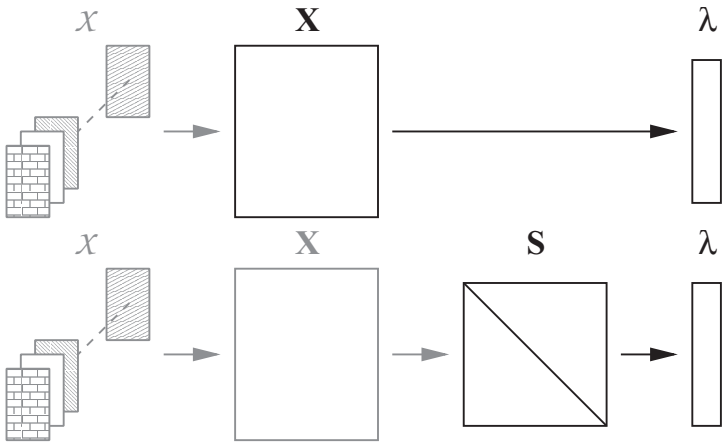


Figure 1.2: Object-centered (top) versus relationship-based clustering (bottom). The focus in this dissertation is on relationship-based clustering which is independent of the feature space.

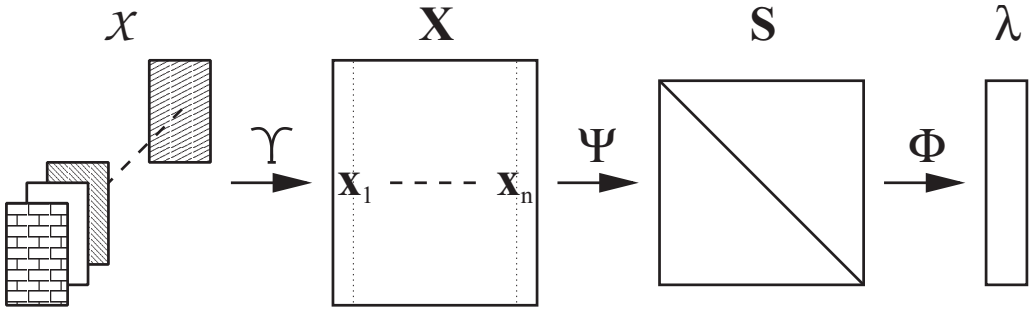


Figure 1.3: Abstract overview of the general relationship-based, single-layer, single-learner, batch clustering process from a set of raw object descriptions $\mathcal{X}$ to the vector of cluster labels λ: $(\mathcal{X} \in \mathcal{I}^n) \xrightarrow{\Upsilon} (\mathbf{X} \in \mathcal{F}^n \subset \mathbb{R}^{d \times n}) \xrightarrow{\Psi} (\mathbf{S} \in \mathcal{S}^{n \times n} = [0,1]^{n \times n} \subset \mathbb{R}^{n \times n}) \xrightarrow{\Phi} (\lambda \in \mathcal{O}^n = \{1, \ldots, k\}^n)$

Finally, λ is computed using e.g., graph partitioning.

1.4 Current Challenges in Clustering

Many traditional clustering techniques [Har75, Nie81, JD88] do not perform satisfactorily in data mining scenarios due to a variety of reasons. These reasons can be divided into those arising from the data distribution and those caused by application constraints:

- **Data Distribution**

  Large number of samples. The number of samples to be processed is very high. Algorithms have to be very conscious of scaling issues. Like many interesting problems, clustering in general is NP-hard, and practical and successful data mining algorithms usually scale linear or log-linear. Quadratic and cubic scaling may also be allowable but a linear behavior is highly desirable.

  High dimensionality. The number of features is very high and may even exceed the number of samples. So one has to face the curse of dimensionality [Fri94].

  Sparsity. Most features are zero for most samples, i.e. the object-feature matrix is sparse. This property strongly affects the measurements of similarity and the computational complexity.

  Strong non-Gaussian distribution of feature values. The data is so skewed that it can not be safely modeled by normal distributions.

Significant outliers. Outliers may have significant importance. Finding these outliers is highly non-trivial, and removing them is not necessarily desirable.

- **Application context**

  Legacy clusterings. Previous cluster analysis results are often available. This knowledge should be reused instead of starting each analysis from scratch.

  Distributed data. Large systems often have heterogeneous distributed data sources. Local cluster analysis results have to be integrated into global models.

For example, in market-basket analysis and text document clustering, we found the number of samples ranging from 10^3 to 10^5, each sample having around 10^3 to 10^5 attributes. On average, over 99% of the attributes are zero, resulting in a very non-Gaussian feature value distribution. In fact, the distribution is often modeled best by a Poisson with a point mass at 0. Outliers are often present and important, such as restaurant owners in grocery shopping records, or index pages in document clustering.

In the document clustering application context, a multitude of legacy clusterings is available from several sources, such as Yahoo!, DMOZ, or Northern Light, which can be exploited for current analysis. The new challenges of high-dimensional, large-scale, heterogeneous databases create the need for new approaches to clustering.

1.5 Contributions

The goal of this dissertation is to improve cluster analysis of complex, high-dimensional, and sparse data, especially when the application scenario imposes constraints on the desired results and on the distribution of and access to the data. This dissertation utilizes ideas from pattern recognition, machine learning, statistics, graph theory, matrix reordering, multi-learner systems, and information theory to build a novel paradigm for cluster analysis based on relationships. The specific contributions of this dissertation are as follows:

- Development of a complete framework for behavioral customer segmentation. The framework extends previous work through domain specific similarity measures such as the extended Jaccard coefficient and constraints such as revenue or customer balancing.

- Proposal of an intuitive and interactive clustering visualization method based on a reordering of the similarity matrix.

- Development of a comparative framework for semi-supervised text clustering and investigation of several popular clustering approaches on a variety of data-sets. The empirical evaluation demonstrates how relationship-based methods improve both quality as well as balance of results.

- Definition of the cluster ensemble problem as a counterpart to classification ensembles in unsupervised learning. The problem of combining previous clusterings without resorting to the original features is posed as a mutual information maximization problem.

- Development and comparison of three relationship-based algorithms for

the cluster ensemble problem. It is demonstrated that all of them work well on real data and are able to deal with missing labels and soft clusterings.

- Application of cluster ensembles to foster robustness and to enable distributed clustering.

1.6 Organization

In the following chapter, background and related work are discussed. Relationship-based clustering and visualization is presented in chapter 3 [SG00c, SG00a, SG00b, SG01, GS02, SG02b]. In chapter 4 [SGM00] we present a comparison over a variety of clustering approaches in document clustering with a novel evaluation framework. In chapter 5 [SG02a], we introduce the cluster ensemble problem and several methods to solve it heuristically. Chapter 6 concludes the dissertation with a summary of the contributions to cluster analysis and presents some future directions of research.

Chapter 2

Background and Related Work

> Few have heard of Fra Luca Pacioli, the inventor of double-entry book-keeping; but he has probably had much more influence on human life than has Dante or Michelangelo.
>
> – Herbert J. Muller[1]

2.1 Overview

Clustering has been widely studied in several disciplines, specially since the early 60's [Har75, JMF99]. Some classic approaches include partitional methods such as k-means, hierarchical agglomerative clustering, unsupervised Bayes, and soft[2] techniques, such as those based on fuzzy logic or statistical mechanics [CG96]. Conceptual clustering [Fis87b], which maximizes category utility, a measure of predictability improvement in attribute values given a clustering, is also popular in the machine learning community. In most clas-

[1] In *The Uses of the Past*, 1952

[2] In *soft* clustering, a record can belong to multiple clusters with different degrees of 'association' [KG99].

sical techniques, and even in fairly recent ones proposed in the data mining community (CLARANS, DBSCAN, BIRCH, CLIQUE, CURE, WAVECLUSTER, etc. [RS99, HKT01]), the objects to be clustered only have numerical attributes and are represented by low-dimensional feature vectors. The clustering algorithm is then based on distances between the samples in the original vector space [SWY75]. Thus these techniques are faced with the 'curse of dimensionality' and the associated sparsity issues when dealing with very high-dimensional data such as text. Indeed, often, the performance of such clustering algorithms is demonstrated only on illustrative 2-dimensional examples.

Clustering algorithms may take an alternative view based on a notion of *similarity* or dissimilarity. Similarity is often derived from the inner product between vector representations, a popular way to quantify document similarity. In [DM01], the authors present a spherical k-means algorithm for document clustering using this similarity measure. Graph-based clustering approaches that attempt to avoid the 'curse of dimensionality' by transforming the problem formulation into a similarity space include [KHK99, BGG$^+$99, SG00c]. Finally, when only pairwise similarities are available, techniques such as Multi-Dimensional Scaling (MDS) [Tor52] have been used to embed such points into a low-dimensional space such that the stress (relative difference between embedded point distances and actual distances) is minimized. Clustering can then be done in the embedding space. However, in document clustering this is not commonly used since for acceptable stress levels the dimensionality of the embedding space is too high.

Clustering has also been studied for the purpose of *browsing*. A 2-dimensional Self-Organizing Map (SOM) [Koh95] has been applied to produce

a map of Usenet postings in WEBSOM [HKLK97]. The emphasis in WEBSOM is not to maximize cluster quality but to produce a human interpretable 2-dimensional spatial map of known categories (e.g., newsgroups). In the Scatter/Gather approach [CKPT92], document clustering is used for improved interactive browsing of large query results. The focus on this work is mostly on speed/scalability and not necessary maximum cluster quality. In [ZE98], clustering effectiveness was studied for its effectiveness on web documents.

There is also substantial work on *categorizing* documents. Here, since at least some of the documents have labels, a variety of supervised or semi-supervised techniques can be used [MR99, NMTM98]. A technique using the support vector machine is discussed in [Joa98]. There are several comparative studies on document classification [YP97, Yan99].

Dimensionality reduction for text classification/clustering has been studied as well. Often, the data is projected onto a small number of dimensions corresponding to principal components or a scalable approximation thereof (e.g., FASTMAP [FL95]). In Latent Semantic Indexing (LSI) [DDL$^+$90] the term-document matrix is modeled by a rank-K approximation using the top K singular values. While LSI was originally used for improved query processing in information retrieval, the base idea can be employed for clustering as well.

In *bag-of-words* approaches, the term-frequency matrix contains occurrence counts of terms in documents. Often, the matrix is preprocessed in order to enhance discrimination between documents. There are many schemes for selecting term and global normalization components. One popular preprocessing is normalized Term Frequency and Inverse Document Frequency (TF-IDF), which also comes in several variants [Sal89, BY99]. However, this dissertation will not discuss the properties of feature extraction, see e.g., [Kol97, Lew92]

instead. In [YP97, Yan99] classification performance of several other preprocessing schemes are compared.

2.2 Clustering Algorithms

This section gives a brief overview over previous work in general purpose clustering algorithms. The most popular and the best understood clustering algorithm is k-means. A robust version of k-means that uses the medoid instead of the mean to assure that a cluster's representative is an actually observed sample is k-medoids. In agglomerative nearest-neighbor clustering, the singleton cluster samples are successively merged into a tree structure in $n-1$ steps. In each step the closest pair, the nearest-neighbors, are joined. All three algorithms generally are intended for use with metric distances. However, especially for our high-dimensional applications it may be useful to replace the metric distances by a more general similarity measure. Also, artificial neural systems and dimensionality reducing techniques have been successfully applied to clustering. In statistical pattern recognition, the data is modeled as a mixture of parametric density functions, which yields a theoretically optimal treatment of the problem. However, the appropriate parametric families are often not known in advance and the growth of the number of parameters raises estimation error (variance) issues. Recently, the database community proposed a variety of highly scalable approaches for large data-sets. However, the models are very simple (usually Euclidean/Gaussian) and may not fit the properties of high-dimensional data. The graph metaphor provides a well-known framework to pose the clustering problem. Efficient partitioning algorithms exist, but theoretical properties for clustering and experimental

evaluations still need to be explored.

2.2.1 The k-means Framework

The k-means algorithm is a very simple and very powerful iterative technique to partition a data-set into k disjoint clusters, where the value k has to be pre-determined [DH73, Har75]. A generalized, similarity-based description of the algorithm can be given as follows:

1. Start at $t = 0$ with k randomly selected objects as the cluster centers $\mathbf{c}_\ell^{(0)}$, $\ell \in \{1, \ldots, k\}$.

2. Assign each object $\mathbf{x}_j$ to the cluster center with maximum similarity:

$$\lambda_j^{(t)} = \arg \max_{\ell \in \{1,\ldots,k\}} s(\mathbf{x}_j, \mathbf{c}_\ell^{(t)}) \tag{2.1}$$

3. Update all k cluster means:

$$\mathbf{c}_\ell^{(t+1)} = \frac{1}{|\mathcal{C}_\ell^{(t)}|} \sum_{\lambda_j^{(t)}=\ell} \mathbf{x}_j \tag{2.2}$$

4. If any $\mathbf{c}_\ell^{(t+1)}$ differs from $\mathbf{c}_\ell^{(t)}$ go to step 2 with $t \leftarrow t+1$ unless termination criteria (such as exceeding the maximum number of iterations) are met.

When similarity is based on a strictly monotonic decreasing mapping of Euclidean distances, k-means greedily minimizes the sum of squared distances of the samples $\mathbf{x}_j$ to the closest cluster centroid $\mathbf{c}_{\lambda_j}$ as given by equation 2.3.

$$\sum_{j=1}^{n} \|\mathbf{x}_j - \mathbf{c}_{\lambda_j}\|_2^2 = \sum_{\ell=1}^{k} \sum_{\mathbf{x} \in \mathcal{C}_\ell} \|\mathbf{x} - \mathbf{c}_\ell\|_2^2 \tag{2.3}$$

2.2.2 Robust k-medoids

Whenever higher robustness [Hub81] is sought or when means are not meaningful (e.g., certain binary features), the k-medoids algorithm, while computationally expensive, might be the method of choice. It uses medoids (representative sample objects) instead of means. However, the extension of a median to multivariate data is usually realized using a randomized approach. A random object is selected and the cost function is evaluated assuming that the selected object is one of the k-medoids. If the cost decreases, a swap is performed and the search is iterated until no changes occur for all medoids.

2.2.3 Agglomerative Nearest-neighbor Clustering

Nearest-neighbor clustering is an agglomerative single-link clustering technique. Starting with each sample as a singleton cluster, at each stage, the closest pair of clusters is merged. After $n-1$ stages, only one cluster remains and the resulting binary tree can be cut at the desired (or user specified) depth to yield a specific partitioning. The central design choice is the definition of cluster distance:

Minimum distance (single-link). The distance of two clusters equals the minimum distance of all pairs of points taken one from each cluster. This definition yields an algorithm similar to the Minimum Spanning Tree (MST) algorithm and is susceptible to 'bridging': a line of points connecting clusters may cause their unintended merge.

Maximum distance (complete-link). Conversely, using the maximum pairwise distance prefers spherical and compact clusters and minimizes clus-

ter diameter.

Average distance. Balancing this trade-off one might also choose the average pair-wise distance as the cluster distance.

Mean distance. An efficient method is to compute the mean of each cluster and define cluster distance as the distance between the two clusters means. Then, not all pairs' distances have to be computed.

However, the $O(n^2)$ to $O(n^3)$ complexity of nearest-neighbor clustering and the greedy nature without any backtracking are significant drawbacks.

2.2.4 Artificial Neural Systems

In artificial neural networks [Bis95, MMR97], the input passes through a connected network of simple processing units called neurons to the output. Often, each neuron represents a function $\Phi : \mathbb{R}^d \to \mathbb{R}$ which maps several inputs to a single output by a weighted summation and non-linear transformation step. In competitive networks (such as Hamming network, MaxNet) a neuron has excitory effect on itself and inhibitory effects on its neighbors. For example, the max-function and k-means clustering can be expressed as a winner-take-all network. An adaptive approach that allows for a variable number of clusters and is controlled by a vigilance parameter is Adaptive Resonance Theory (ART) [CG88]. The most popular networks for clustering are topologically organized networks. The Self-Organizing (feature) Map (SOM) was proposed by Kohonen [Koh95] and fits a predefined network structure to the data in a topology preserving fashion by updating the winner *and* its adjacent neighbors.

2.2.5 Projective Techniques

In many domains, the high-dimensional representation contains redundant information considering the clustering task. In document clustering, for example, many different words and phrases can be used to convey a similar meaning. Hence, the vector-space representation contains highly correlated evidence and the clustering process can be extended as follows:

1. Find an appropriate projection of the original data ($D > 10^5$) into concept space ($d > 10^2$).

2. Perform clustering in the dimensionality reduced space.

Good sub-spaces for projections are characterized by preserving most of the 'information' in the data. For a regular matrix the K eigenvectors with the K highest eigenvalues are the K orthogonal directions of projection that explain the maximum possible of the variance in the data [Str88]. The extension of the eigen decomposition to non-square matrices is the Singular Value Decomposition (SVD). SVD is closely related to Principal Component Analysis (PCA) which is based on the SVD of the zero-mean normalized data. In Latent Semantic Indexing (LSI) the $d \times n$ word-document matrix $\mathbf{X}$ is modeled by a rank-K approximation using the top K singular values. While LSI is originally used for improved query processing in information retrieval the base idea may be employed for clustering as well. A matrix $\mathbf{U}$ is unitary if and only if $\mathbf{U}^\dagger\mathbf{U} = \mathbf{I}$. Equation 2.4 gives the singular value decomposition (SVD) of the $d \times n$ data matrix $\mathbf{A} = \mathbf{X}$ into a product of the unitary $d \times r$ matrix $\mathbf{U}$, the diagonal $r \times r$ matrix $\mathbf{\Sigma}$, and the unitary $r \times n$ matrix $\mathbf{V}^\dagger$.

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\dagger,\ \mathbf{U}^\dagger\mathbf{U} = \mathbf{I},\ \mathbf{V}^\dagger\mathbf{V} = \mathbf{I},\ \forall(i \neq j) : \sigma_{i,j} = 0 \tag{2.4}$$

r is the rank of $\mathbf{X}$. $\mathbf{A}_K$ is the best rank K-approximation of $\mathbf{A}$ (with $K < r$) which is obtained by using $\mathbf{\Sigma}_K$, which is $\mathbf{\Sigma}$ with the original upper-left K diagonal entries and all others set to 0 (equation 2.5)

$$\mathbf{A}_K = \mathbf{U}_K \mathbf{\Sigma}_K \mathbf{V}_K^\dagger = \sum_{i=1}^{K} \sigma_i \mathbf{u}_i \mathbf{v}_i^\dagger \tag{2.5}$$

The left-singular column vectors $\mathbf{u} \in \mathbb{R}^d$ in $\mathbf{U}$ are called the concept vectors with $\|\mathbf{u}\| = 1$. In general, $\mathbf{A}$ looses its sparsity since the projection is not axis-parallel. However, the concept space does not have to be instantiated explicitly, but can be rather stored functionally, as for example, a product of two sparse matrices. In the vector space model [SWY75, Kol97], a keyword search can be expressed as a matrix product as shown in 2.6. A binary valued query vector $\mathbf{q}$ is projected linearly to a match vector $\mathbf{p}$.

$$\mathbf{p} = \mathbf{A}^\dagger \mathbf{q} \tag{2.6}$$

In the context of web-search, it has been claimed that projecting the query vector and the web-page vectors to concept space outperforms keyword search. The rationale for this observation lies in the fact that the rank-K approximation is actually more representative than the original data matrix, because noise and redundancy have been removed. Also, it has been argued that by reducing the matrix complexity to a small number of concepts, implicitly the query is extended to encompass redundancy as introduced e.g., by synonyms. Using the optimal rank-K-approximation (LSI) yields equation 2.7, where $\mathbf{A}_K$ is given by equation 2.5.

$$\hat{\mathbf{p}} = \mathbf{A}_K^\dagger \mathbf{q} = (\mathbf{V}_K \mathbf{\Sigma}_K)(\mathbf{U}_K \mathbf{q}) \tag{2.7}$$

Generally, the rank of A has a magnitude of 10^3 to 10^5 and K is around 10^2. Recently, it has been reported that random subspace projections perform

very well for high-dimensional data and are close to the optimal projection as given by the SVD [SS97].

2.2.6 Mixture Density Estimation

In statistical pattern recognition, the data is considered as a set of observations from a parametric probability distribution. [Fuk72, DH73]. In a two stage process, the parameters $\mathbf{\Theta}$ of the relevant distributions are *learned* and later applied to *predict* the behavior or origin of a new observation. In Maximum Likelihood (ML) estimation, the parameters $\hat{\mathbf{\Theta}}$ are chosen such that the probability of the observed samples $\mathbf{X}$ is maximized.

$$\hat{\mathbf{\Theta}} = \arg\max_{\mathbf{\Theta}} p(\mathbf{X}|\mathbf{\Theta}) \tag{2.8}$$

Assuming that all samples are pairwise independent yields

$$p(\mathbf{X}|\mathbf{\Theta}) = \prod_{j=1}^{n} p(\mathbf{x}_j|\mathbf{\Theta}) \tag{2.9}$$

Since each sample is drawn from a mixture distribution [EH81, Pri94] we have

$$p(\mathbf{x}_j|\mathbf{\Theta}) = \sum_{\ell=1}^{K} P(\omega_\ell) \cdot p(\mathbf{x}_j|\mathbf{\Theta}, \omega_\ell),\ \sum_{\ell=1}^{K} P(\omega_\ell) = 1 \tag{2.10}$$

The cluster-conditional probability may be assumed to be a multivariate Gaussian which is defined as follows

$$p(\mathbf{x}_j|\mathbf{\Theta}, \omega_\ell) = p(\mathbf{x}_j|\mu_\ell, \mathbf{\Sigma}_\ell) = \frac{1}{\sqrt{(2\pi)^d |\mathbf{\Sigma}_\ell|}} e^{-\frac{1}{2}(\mathbf{x}_j - \mu_\ell)^\dagger \mathbf{\Sigma}_\ell^{-1} (\mathbf{x}_j - \mu_\ell)} \tag{2.11}$$

If there is domain knowledge or a desired behavior of the parameter $\mathbf{\Theta}$'s distribution, Bayes' learning should be used instead of the ML estimation.

$$\hat{\mathbf{\Theta}} = \arg\max_{\mathbf{\Theta}} p(\mathbf{\Theta}|\mathbf{X}) \tag{2.12}$$

Again, we expand $p(\mathbf{\Theta}|\mathbf{X})$ using Bayes' rule.

$$p(\mathbf{\Theta}|\mathbf{X}) = \frac{p(\mathbf{\Theta}) \cdot p(\mathbf{X}|\mathbf{\Theta})}{p(\mathbf{X})} \tag{2.13}$$

The prior distribution $p(\mathbf{\Theta})$ can be known from the domain or estimated, $p(\mathbf{X}|\mathbf{\Theta})$ is the ML estimate as described above and $p(\mathbf{X})$ can be ignored for optimization purposes since it is constant in respect to $\mathbf{\Theta}$.

The learned distributions $p(\mathbf{x}|\mathbf{\Theta}, \omega_\ell)$ can now be used for categorization and prediction of a sample's cluster label. The Bayes classifier is optimal in terms of prediction error, assuming that the distribution of the data is known precisely:

$$\lambda_j = \arg \max_{\ell \in \{1,\ldots,K\}} (P(\omega_\ell) \cdot p(\mathbf{x}_j|\mathbf{\Theta}, \omega_\ell)) \tag{2.14}$$

Often, using the log-likelihood (equation 2.15) instead of the actual probability values has advantages for optimization (e.g., convexity, products of very small probabilities which may be problematic for fixed precision numerics are avoided).

$$l(\mathbf{X}) = -\log p(\mathbf{X}) \tag{2.15}$$

The theory behind statistical models is very well understood and explicit computations of error bounds are advantageous. Statistical formulations are advantageous for soft clustering problems with a moderate number of dimensions d. The very powerful Expectation-Maximization (EM) algorithm [DLR77] has been applied to k-means [FRB98]. However, these parametric models tend to impose structure on the data, that may not be there. The selected distribution family may not be really appropriate. In fact, high-dimensional data as found in data mining is distributed strongly non-Gaussian. Also, the number of parameters increases rapidly with d so that the estimation problem becomes more and more ill-posed. Non-parametric models, like

v-nearest-neighbor, have been found preferable in many tasks where a lot of data is available.

2.2.7 Recent Database-driven Approaches

In BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [ZRL97], the authors propose to incrementally build a balanced tree representing the data. Each node $\mathcal{C}_\ell$ holds k Cluster-Features (CF), defined as the triplet of the three sufficient statistics $n_\ell = |\mathcal{C}_\ell|$, $\mathrm{LS}_\ell = \sum_{\mathbf{x}_j \in \mathcal{C}_\ell} \mathbf{x}_j$, and $\mathrm{SS}_\ell = \sum_{\mathbf{x}_j \in \mathcal{C}_\ell} \mathbf{x}_j^2$. These zeroth, first and second statistical moments are sufficient to compute centroid (mean), radius (isotropic standard deviation), diameter (average pairwise intra-cluster Euclidean distance) at any time during the incremental algorithm. The user parameters are the branching factor (maximum number of children) and a splitting threshold on the diameter of a cluster. After building this tree, a clustering algorithm of the users choice is used to cluster the leaf nodes of the BIRCH tree.

CURE (Clustering Using REpresentatives) [GRS98] uses a fixed number of well-scattered representatives for each cluster. This allows the algorithm to adopt the middle ground between storing only the centroid and retaining all samples. Thus the representation is non-parametric, which allows for non-Gaussian distributions. A shrinking factor increases robustness by scaling all samples around the corresponding cluster-center which reduces the effects of outliers.

In CLARANS (Clustering Large Applications based upon RANdomized Search) [NH94], the authors propose to cluster on a randomly selected sub-set of the data using a k-medoids based algorithm. Subsequently, neighboring

solutions (a clustering with one replaced medoid) are explored and the sub-sample is re-randomized until a local optimum is found. The entire process is repeated to de-localize the search and find a more global solution.

When sampling, the success depends strongly on the size of the sub-set, which has to be big enough to preserve the knowledge from the original data. In large database applications, random sub-sampling is often as expensive as performing an entire scan, since the used data structures are not tuned for random access.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [EKSX96] is based on the notion of core objects (defined by having a minimum number of points within an ϵ-neighborhood), density-reachability (non-symmetric), and density-connectedness (symmetric). However, considering the high number of dimensions in data mining applications (and consequently the sparseness of data in feature space) it seems questionable if the notion of density remains meaningful.

2.2.8 Graph-based Clustering

ROCK (Robust Clustering using linKs) [GRS99] is an agglomerative hierarchical clustering technique for categorical attributes. It uses the binary Jaccard coefficient and a thresholding criterion to establish links between samples. The links are unweighted edges in a graph with vertices corresponding to the objects to be clustered. Common neighbors are used to define inter-connectivity of clusters which is used to merge clusters. Another key idea in ROCK is to define a transitive neighbor relationship. Instead of only using the simple links (adjacency matrix $\mathbf{A}$), all pairs with a common neighbor are linked as well (us-

ing **AA**). However, we feel that this does not really add any new information and a good clustering algorithm would not be improved by this modification of the adjacency matrix. In adds redundancy by adding more links and hence the desired sparsity reduced.

CHAMELEON [KHK99] starts with partitioning the data into a large number of clusters by partitioning the v-nearest neighbor graph. In the subsequent stage clusters are merged based on inter-connectivity and their relative closeness.

2.2.9 Graph and Hypergraph Partitioning

The objects to be clustered can be viewed as a set of vertices $\mathcal{V}$. Two webpages $\mathbf{x}_a$ and $\mathbf{x}_b$ (or vertices v_a and v_b) are connected with an undirected edge of positive weight $s(\mathbf{x}_a, \mathbf{x}_b)$, or $(a, b, s(\mathbf{x}_a, \mathbf{x}_b)) \in \mathcal{E}$. The cardinality of the set of edges $|\mathcal{E}|$ equals the number of *non-zero* similarities between all pairs of samples. A set of edges whose removal partitions a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ into k pair-wise disjoint sub-graphs $\mathcal{G}_\ell = (\mathcal{V}_\ell, \mathcal{E}_\ell)$, is called an edge separator. Our objective is to find such a separator with a minimum sum of edge weights. While striving for the minimum cut objective, the number of objects in each cluster has to be kept approximately equal. In this particular case of balancing, the problem is NP-hard and known as *graph partitioning*.

Balanced clusters are desirable because each cluster represents an equally important share of the data. However, some natural classes may not be equal size. By using a higher number of clusters we can account for multi-modal classes (e.g., XOR-problem) and clusters can be merged at a latter stage. The most expensive step in this $O(n^2 \cdot d)$ technique is the computation of the $n \times n$

similarity matrix. In document clustering, sparsity can be induced by looking only at the v strongest edges or at the subgraph induced by pruning all edges except the v nearest-neighbors for each vertex. Sparsity makes this approach feasible for large data-sets.

The Kernighan-Lin (KL) algorithm has been a very successful $O(n^3)$ algorithm for graph partitioning [KL70]. The KL is based on iteratively conducting best sequences of swaps involving all vertices. In an improved implementation by Fiduccia and Mattheyses [FM82], the complexity was of the KL algorithm was reduced to $O(|\mathcal{E}|)$.

In *spectral bisection* [HL95, PSL90] the partitioning problem is reduced to finding the second least dominant eigenvector of the Laplacian matrix ∇. Consider the incidence matrix $\mathbf{C}$ whose entries are defined as follows:

$$c_{i,j} = \begin{cases} 1 & \text{if} \quad (i,a) = \mathbf{e}_j \in \mathcal{E} \\ -1 & \text{if} \quad (a,i) = \mathbf{e}_j \in \mathcal{E} \\ 0 & \text{else} \end{cases} . \tag{2.16}$$

The Laplacian ∇ can be written as

$$\nabla = \mathbf{C}\mathbf{C}^{\dagger} \tag{2.17}$$

and has the vertex degrees on the diagonal. Off-diagonal entries indicate -1 if an edge between the row- and column-vertex exists and 0 otherwise. By construction, ∇ is symmetric and thus has real eigenvalues and its eigenvectors are real and orthogonal. Moreover, all columns and rows sum up to 1. Consequently the smallest eigenvalue is 0 with eigenvector $\mathbf{1}$, or $\nabla\mathbf{1} = 0 \cdot \mathbf{1}$. For graph bi-section ($k = 2$), it turns out that the ratio-cut objective is maximized when splitting according to the signs of the entries in the second smallest eigenvector

of the Laplacian, which is also called the Fiedler vector [Fie73, Fie75]. This algorithm can be extended to $k > 2$ and weighted edges.

Currently, the best available algorithms use the KL-algorithm or spectral bisection in a *multi-level framework* which has three stages:

- Coarsen graph by collapsing appropriate vertices
- Initial partitioning of simplified graph
- Un-coarsen graph and refine partitioning

A hypergraph is a graph whose edges can connect more than two vertices (hyperedges). The clustering problem is then formulated as a finding the minimum-cut of a hypergraph. A minimum-cut is the removal of the set of hyperedges (with minimum edge weight) that separates the hypergraph into k unconnected components. Again, an object $\mathbf{x}_j$ maps to a vertex v_j. Each feature maps to a hyperedge connecting all vertices with a non-zero value for this particular feature, so $|\mathcal{E}| = d$. The minimum-cut of this hypergraph into k unconnected components gives the desired clustering. Hypergraphs are often used in VLSI design. An application to association rule hypergraph clustering can be found in [BGG+99].

A variety of packages for graph partitioning is available. A Matlab kit for geometric mesh partitioning (requires coordinate information for vertices) [GMT95] and spectral bi-section [PSL90] by John R. Gilbert and Shang-Hua Teng is available from Xerox. Currently, the most popular programs for graph partitioning are CHACO (available from Bruce Hendrickson at Los Alamos National Labs) [HL94] and METIS (George Karypis, University of Minnesota) [KK98a, KK98b]. Both implement the currently fastest available algorithm, a

multi-level version of Kernighan-Lin [KL70, FM82]. A popular package for hypergraph partitioning is HMETIS (George Karypis, University of Minnesota) [KAKS97].

2.3 Scalability

High-dimensional data is often very sparse, e.g. most entries in the data matrix are zero. Using this sparsity in efficient data structures and algorithms can reduce temporal and storage complexity by several orders of magnitude. Scalability can be investigated in terms of the number of objects n and the number of dimensions d. Traditionally, scaling to large n has been considered more. In this dissertation, we focus on applications with large d. For scaling to large n, there are four main ways of reducing complexity in order to scale clustering algorithms:

Sampling. Sample the data, cluster the sample points and then use a quick heuristic to allocate the non-sampled points to the initial clusters. This approach will yield a faster algorithm at the cost of some possible loss in quality, and is employed, for example in the Buckshot algorithm for the Scatter/Gather approach to iterative clustering for interactive browsing [CKPT92]. If the sample is $O(\sqrt{n})$, and 'nearest cluster center' is used to allocate the remaining points, one obtains an $O(kn)$ algorithm. Also related are randomized approaches that can partition a set of points into two clusters of comparable size in sublinear time, producing a $(1+\epsilon)$ solution with high probability [Ind99]. We shall show later that since OPOSSUM is based on balanced clusters, sampling is a good choice since one can ensure with high probability that each cluster is represented in

the sample without needing a large sample size.

Sequential building. Construct a 'core' clustering using a small number of elements, and then sequentially scan the data to allocate the remaining inputs, creating new clusters (and optionally adjusting existing centers) as needed. Such an approach is seen e.g., in BIRCH [ZRL97]. This style compromises balancing to some extent, and the threshold determining when a new cluster is formed has to be experimented with to bring the number of clusters obtained to the desired range. A version of this approach for graph partitioning using a corrupted clique model was proposed by [BDSY99] and applied to clustering gene expressions. This can be readily used for OPOSSUM as well. Sequential building is specially popular for out-of-core methods, the idea being to scan the database once to form a summarized model (for instance, the size, sum and sum-squared values of each cluster [BFR98]) in main memory. Subsequent refinement based on summarized information is then restricted to main memory operations without resorting to further disk scans.

Representatives. Compare with representatives rather than with all points. Using $m < n$ representatives reduces the number of similarities to be considered from $O(n^2)$ to $O(nm)$. For example, in k-means, the current cluster means are used as representatives. Since points do not have to compared to all others but only to a few centroids (the current means), scalability is considerably improved. The results, however, become sensitive to the initial selection of representatives. Also, representatives might have to be updated resulting in an iterative algorithm.

Pre-segmentation. Apply prior domain knowledge to pre-segment the data,

e.g. using indices or other 'partitionings' of the input space. Pre-segmentations can be coarser (e.g., to reduce pairwise comparisons by only comparing within segments) or finer (e.g., to summarize points as a preprocessing step as in BIRCH) than the final clustering. As mentioned earlier, this becomes increasingly problematic as the dimensionality of the input space increases to the hundreds or beyond, where the suitable segments may be difficult to estimate, pre-determine, or populate.

All these approaches are somewhat orthogonal to the main clustering routine in that they can be applied in conjunction with most core clustering routines to save computation, at the cost of some loss in quality.

2.4 Visualization

Visualization of high-dimensional data clusters can be largely divided into four popular approaches:

1. Dimensionality reduction by selection of 2 or 3 dimensions, or, more generally, projecting the data down to 2 or 3 dimensions. Often these dimensions correspond to principal components or a scalable approximation thereof (e.g., FASTMAP [FL95]). Chen, for example, creates a browsable 2-dimensional space of authors through co-citations [Che99]. Another noteworthy method is CViz [DMS98], which projects onto the plane that passes through three selected cluster centroids to yield a 'discrimination optimal' 2-dimensional projection. These projections are useful for a medium number of dimensions, i.e., if d is not too large ($<$ 100).[3] Nonlinear projections have also been studied [CG01]. Recreating

[3]For text mining, linearly projecting down to about 20-50 dimensions does not affect

a 2- or 3-dimensional space from a similarity graph can also be done through multi-dimensional scaling [Tor52].

2. Parallel axis plots show each object as a line along d parallel axis. However, this technique is rendered ineffective if the number of dimensions d or the number of objects gets too high.

3. Kohonen's Self Organizing Map (SOM) [Koh90] provides an innovative and powerful way of clustering while enforcing constraints on a logical topology imposed on the cluster centers. If this topology is 2-dimensional, one can readily 'visualize' the clustering of data. Essentially a 2-dimensional manifold is mapped onto the (typically higher dimensional) feature space, trying to approximate data density while maintaining topological constraints. Since the mapping is not bijective, the quality can degrade very rapidly with increasing dimensionality of feature space, unless the data is largely confined to a much lower order manifold within this space [CG01]. Multi-Dimensional Scaling (MDS) and associated methods also face similar issues.

4. Visualization can also be done by showing the data matrix as an image by converting entries to brightness values. The ordering of data points for visualization has previously been used in conjunction with clustering in different contexts. For example, in OPTICS [ABKS99] instead of producing an explicit clustering, an augmented ordering of the database is produced. Subsequently, this ordering is used to display various metrics

results much (e.g., LSI). However, it is still too high to visualize. A projection to lower dimensions leads to substantial degradation and 3-dimensional projections are of very limited utility.

such as reachability values. In cluster analysis of genome data [ESBB98] re-ordering the primary data matrix and representing it graphically has been explored. This visualization takes place in the primary data space rather than in the relationship-space. Sparse primary data matrix re-orderings have also been considered for browsing hypertext [BHR96].

A useful survey of visualization methods for data mining in general can be found in [KK96]. The popular books by E. Tufte [Tuf83] on visualizing information are also recommended.

2.5 Ensembles and Knowledge Reuse

There is an extensive body of work on combining multiple classifiers or regression models, but little on combining multiple clusterings so far in the machine learning community. However, in traditional pattern recognition, there is a substantial body of largely theoretical work on *consensus classification* during the mid-80's and earlier [NN86a, NN86b, BLM86]. These studies used the term 'classification' in a very general sense, encompassing partitions, dendrograms and n-trees as well. Today, such operations are typically referred to as clusterings. In consensus classification, a profile is a set of classifications which is sought to be integrated into a single consensus classification. A representative work is that of [NN86a], which investigated techniques for strict consensus. They first construct a lattice over the set of all partitionings by using the refinement relation. Partitioning A is a refinement of partitioning B if every cluster in A is a subset of some cluster in B. This refinement relation defines a partial order, and thus for each pair of partitionings, a supremum and an infimum exist. A strict consensus finds the supremum and infimum of

all given pairs of clusterings, thus yielding the consensus interval.

Such work on strict consensus works well for small data-sets with little noise and little diversity. But, in presence of strong noise the results can be trivial, namely the supremum is the monolithic clustering (one cluster) and the infimum is the set of singletons. Also, the computations are intractable for large data-sets. Another drawback is that the strict consensus is not at the same level of resolution.

The most prominent application of strict consensus is by the computational biology community to obtain phylogenetic trees [KW99, KWY95]. A set of DNA sequences can be used to generate evolutionary trees using criteria such as maximum parsimony, but often one obtains several hundreds of trees with the same score function. In such cases, biologists look for the strict *consensus tree*, the 'infimum', which has lower resolution but is compatible with all the individual trees. Note that such systems are different from the cluster ensembles proposed in chapter 5 in that (i) they are hierarchical clusterings, typically using *unrooted trees*, (ii) have domain specific distance metrics (e.g., Robinson-Foulds distance) and evaluation criteria such as parsimony, specificity and density, and (iii) *strict* consensus is a requirement.

The most important difference between our work in chapter 5 and studies of the 80's on 'consensus classification' is that the latter obtains consensus at a *different* level of refinement, whereas this dissertation focuses on consensus at the *same* level or scale. For example, taking the intersections of two partitionings results in a number of partitions up to the product of the numbers of partitions in the original partitionings, thus moving the analysis to a much finer scale.

Two interesting applications of using consensus ideas to help classifi-

cation/regression problems have emerged recently. Consensus decision trees have been used to generate a single decision tree from N-fold cross-validated C4.5 results [KLF01]. Objects are clustered according to their positions or paths in the N decision trees. Then, only objects of the majority class in each cluster are selected to form a new training set that generates the consensus decision tree. The goal here is to obtain a single, simplified decision tree without compromising much on classification accuracy. In [Rag01], many models are obtained for a regression problem. These models are then clustered, based on their pairwise mutual information. Finally, a smaller committee is obtained for the regression problem by selecting a representative model from each cluster. Note that neither of these works actually combine multiple partitionings of the input data, but instead use clustering as an intermediate step in solving a classification or regression problem.

In chapter 5, we will propose to exploit multiple existing groupings of the data. Several analogous approaches exist in supervised learning scenarios (class labels are known), under categories such as 'life-long learning' [Thr96], 'learning to learn' [TP97] and 'knowledge reuse' [BG98, BG99]. Several researchers have attempted to directly reuse the internal state information from classifiers under the belief that related classification tasks may benefit from common internal features. One approach to this idea is to use the weights of hidden layers in an Multi-Layer Perceptron (MLP) classifier architecture to represent the knowledge to be shared among the multiple tasks being trained on simultaneously [Car95]. Pratt [Pra94] uses some of the trained weights from one MLP network to initialize weights in another MLP to be trained for a later, related task. In a related work, Silver and Mercer [SM96] have developed a system consisting of *task* networks and an *experience* network. The experience

network tries to learn the converged weights of related task networks in order to initialize weights of target task networks. Both of these weight initialization techniques resulted in improved learning speed. Besides weight reuse in MLP type classifiers, other state reuse methods have been developed. One approach, developed by Thrun [TO96], is based on a nearest neighbor classifier in which each of the dimensions of the input space is scaled to bring examples within a class closer together while pushing examples between different classes apart. The scaling vector derived for one classification task is then used in another, related task. We have previously proposed a knowledge reuse framework wherein the labels produced by old classifiers are used to improve the generalization performance of a new classifier for a different but related task [BG98]. This improvement is facilitated by a supra-classifier that accesses only the outputs of the old and new classifiers, and does not need the training data that was used to create the old classifiers. Substantial gains are achieved when the training set size for the new problem is small, but can be compensated for by the extraction of information from the existing related solutions.

More recently, a host of semi-supervised methods have emerged that augment a limited training set of labeled data by a larger amount of unlabelled data. One powerful idea is to use *co-training* [BM98], whose success hinges on the presence of multiple 'redundantly sufficient' views of the data. For example, Muslea et al. introduced a multi-view algorithm including active sampling based on co-training [MMK01]. Nigam and Ghani investigated the effectiveness of co-training in semi-supervised settings [NG00].

Another application of our cluster ensembles is to combine multiple clustering that were obtained based on only partial sets of features. This problem has been approached recently as a case of collective data mining [KPHJ99]. In

[JK99] a feasible approach to combining distributed agglomerative clusterings is introduced. First, each local site generates a dendrogram. The dendrograms are collected and pairwise similarities for all objects are created from them. The combined clustering is then derived from the similarities. In [KHSJ01], a distributed method of principal components analysis is introduced for clustering.

The usefulness of having multiple views of data for better clustering has been recognized by others as well. In multi-aspect clustering [MS00], several similarity matrices are computed separately and then integrated using a weighting scheme. Also, Mehrotra has proposed a multi-viewpoint clustering, where several clusterings are used to semi-automatically structure rules in a knowledge base [Meh99].

In our proposed algorithms, we use *hypergraph* representations which have been extensively studied [GJ79]. Hypergraphs have been previously used for (a single) high-dimensional clustering [HKKM97] but not for combining multiple groupings. *Mutual information* [CT91] is a useful measure in a variety of contexts. For example, the information bottleneck method [ST00] is an information-theoretical approach that uses mutual information to do dimensionality reduction (e.g., through clustering) while trying to preserve as much information about the class labels as possible.

2.6 Challenges

In section 1.4, we introduced some of the challenges clustering faces in current data mining scenarios. In this section, we highlight a selection of challenges that have been identified and addressed more extensively in previous work.

2.6.1 The Problem of Scale

Most clustering algorithms assume the number of clusters to be known *a priori*. The desired granularity is generally determined by external, problem specific criteria. For example, such a criterion might be the user's willingness to deal with complexity and information overload. This issue of scale remains mostly unsolved although various promising attempts such as Occam's razor, minimum description length, category utility, etc. have been made.

Finding the 'right' number of clusters, k, for a data-set is a difficult, and often ill-posed, problem. Gelman in [GCSR95] on page 424 says in a mixture modeling context: "One viewpoint is that the problem of finding the best number of clusters is fundamentally ill-defined and best avoided." Let us illustrate the problem by a data-set where points are arranged in a grid-like pattern on the 2D plane (figure 2.1). The points are arranged in four squares such that when 'zooming in' the same square structure exists but on a lower scale. When asking what is the number of clusters for that data-set, there are at least 3 good answers possible, namely 4, 16, and 64. Depending on the desired resolution the 'right' number of clusters changes. In general, there are multiple good k's for any given data. However, a certain number of clusters might be more stable than others. In the example shown in figure 2.1, five clusters is probably not a better choice than four.

Since there seems to be no definite answer to how many clusters are in a data-set, a user-defined criterion for the resolution has to be given instead. In the general case, the number of clusters is assumed to be known. Alternatively, one might want to reformulate the specification of scale through an upper bound of acceptable error (which has to be suitably defined) or some

other criterion. Many heuristics for finding the right number of clusters have been proposed. Occam's razor states that the simplest hypothesis that fits the data is the best [BEHW87, Mit97]. Generally, as the model complexity grows the fit improves. However, over-learning with a loss of generalization may occur. In probabilistic clustering, likelihood-ratios, v-fold-cross-validation, penalized likelihoods (e.g., Bayesian information criterion), and Bayesian techniques (AUTOCLASS) are popular [Smy96, MC85]:

Cross-validation. A recent Monte Carlo cross-validation based approach [Mil81] which minimizes the Kullback-Leibler information distance to find the right number of clusters can be found in [Smy96].

Category Utility. In machine learning, category utility [GC85, Fis87a] has been used to asses the quality of a clustering for a particular k. Category utility is defined as the *increase* in the expected number of attribute values $v_{i,h}$ (the h-th discrete level of feature i) that can be correctly guessed, given a partitioning λ *over* the expected number of correct guesses with no such knowledge. A weighted average over categories allows comparison of different size partitions.

Bayesian Solution. In the full Bayesian solution the posterior probabilities of all values of k are computed from the data and priors for the data and k itself. AUTOCLASS [CS96] uses this approach with some approximations to avoid computational issues with the complexity of this approach. The complexity makes this approach infeasible for very high-dimensional data.

Regularization. A feasible approach for high-dimensional data mining is a

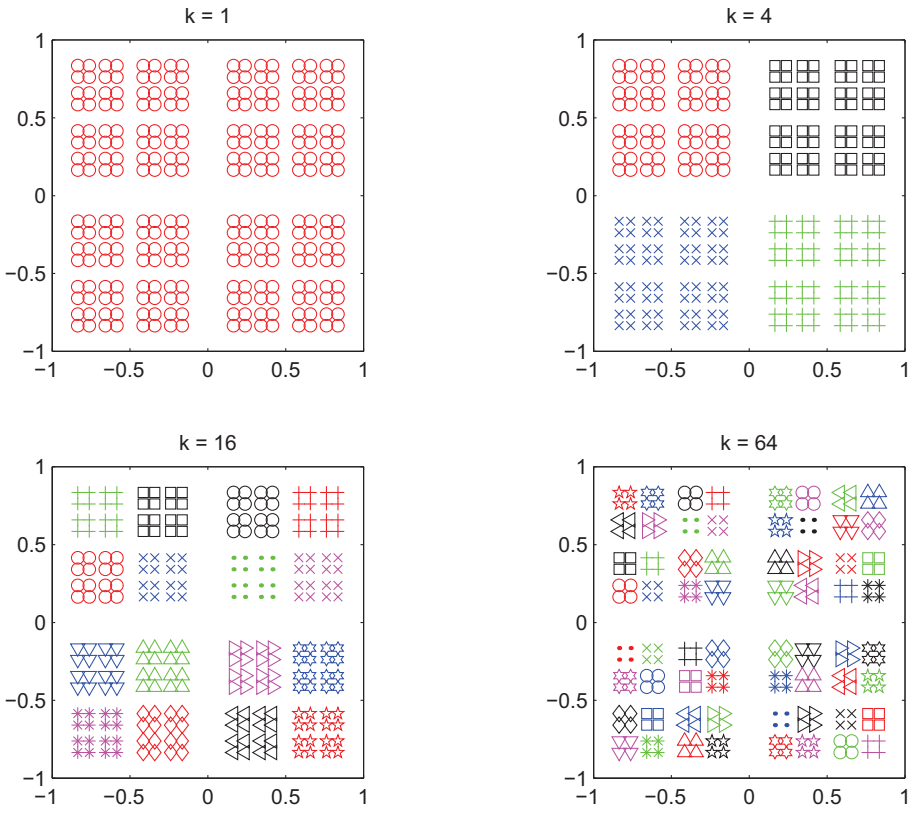


Figure 2.1: Example for the problem of scale. Four examplary clusterings are shown for the 2-dimensional `RECSQUARE` data-set. 4, 16, and 64 are 'equally good' choices for the number of clusters k.

regularization-based approach. A penalty term is introduced to discourage a high number of parameters. Variations of this theme include penalized likelihoods, such as the Bayesian Information Criterion (BIC), and coding-based criteria, such as Minimum Description Length (MDL).

2.6.2 Curse of Dimensionality

When dealing with very high-dimensional data, one is faced with the 'curse of dimensionality' [Fri94] and the associated sparsity issues. Essentially the amount of data to sustain a given spatial density increases exponentially with the dimensionality of the input space, or alternatively, the sparsity increases exponentially given a constant amount of data, with points tending to become equidistant from one another. In general, this will adversely impact any method based on spatial density, unless the data follows certain simple distributions. Figure 2.2 gives a simple illustration of the curse of dimensionality.

2.6.3 Clustering Objectives

The informal description of clustering as finding meaningful groups does not suggest a straightforward way of evaluating clusters or defining an objective function. Many definitions of good clustering exist. Prominently, clustering has been posed as an optimization problem for minimum error (least squared error) or maximum attribute predictability (category utility). However, many proposals have been made to evaluate cluster quality or validity [JMF99]. No consensus has been reached in the community and this dissertation will give a brief overview of evaluation criteria in section 4.4.

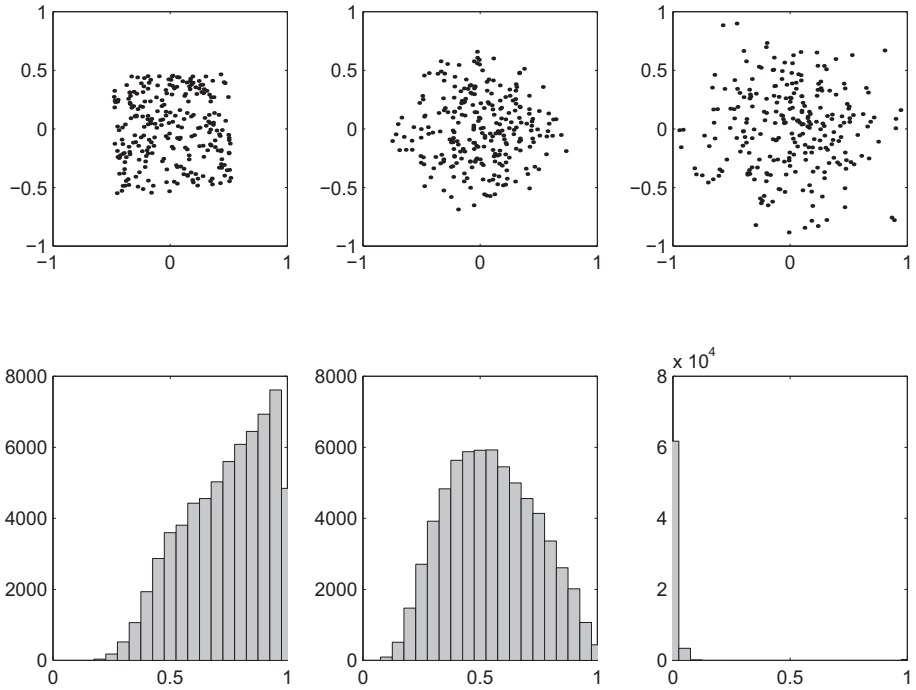


Figure 2.2: Curse of dimensionality illustrated with 256 d-dimensional points from a [0,1] uniform distribution with $d = 2$ (left), 4 (middle) and 32 (right). The top row shows the results of the 2D Principal Components Analysis (PCA). The bottom row shows how similarity (as a monotonically decreasing function of Euclidean distance) is distributed. As d increases, projections approach Gaussian distributions. Also, an average pair of points' similarity decreases rapidly and similarities become approximately equal for most pairs with increasing d.

Chapter 3

Relationship-based Clustering and Visualization

One picture is worth ten thousand words.

– Frederick R. Barnard[1]

In several real-life data mining applications, data resides in *very high* (1000 and more) dimensional space, where both clustering techniques developed for low dimensional spaces (k-means, BIRCH, CLARANS, CURE, DBScan, etc.) as well as visualization methods, such as parallel coordinates or projective visualizations, are rendered ineffective. This chapter proposes a relationship-based approach that alleviates both problems, side-stepping the 'curse of dimensionality' issue by working in a suitable similarity space instead of the original high-dimensional attribute space. This intermediary similarity space can be suitably tailored to satisfy business criteria such as requiring customer clusters to represent comparable amounts of revenue. We apply effi-

[1]In *Printers' Ink*, March 1927

cient and scalable graph partitioning-based clustering techniques in this space. The output from the clustering algorithm is used to re-order the data points so that the resulting permuted similarity matrix can be readily visualized in 2 dimensions, with clusters showing up as bands. While 2-dimensional visualization of a similarity matrix is by itself not novel, its combination with the order-sensitive partitioning of a graph that captures the relevant similarity measure between objects provides three powerful properties: (i) the high-dimensionality of the data does not affect further processing once the similarity space is formed; (ii) it leads to clusters of (approximately) equal importance, and (iii) related clusters show up adjacent to one another, further facilitating the visualization of results. The visualization is very helpful for assessing and improving clustering. For example, actionable recommendations for splitting or merging of clusters can be easily derived, and it also guides the user towards the right number of clusters. Results are presented on a real retail industry data-set of several thousand customers and products, as well as on clustering of web-document collections and of web-log sessions.

3.1 Motivation

Knowledge discovery in databases often requires clustering the data into a number of distinct segments or groups in an effective and efficient manner. Good clusters show high similarity within a group and low similarity between any two different groups. Besides producing good clusters, certain clustering methods provide additional useful benefits. For example, Kohonen's Self-Organizing feature Map (SOM) [Koh90] imposes a logical, 'topographic' *ordering* on the cluster centers such that centers that are nearby in the logical

ordering represent nearby clusters in the feature space. A popular choice for the logical ordering is a two-dimensional lattice which allows all the data points to be projected onto a two-dimensional plane for convenient visualization [Hay99]. While clustering is a classical and well studied area, it turns out that several data mining applications pose some unique challenges that severely test traditional techniques for clustering and cluster visualization. For example, consider the following two applications:

- Grouping customers based on buying behavior to provide useful marketing decision support knowledge; especially in e-business applications where electronically observed behavioral data is readily available. Customer clusters can be used to identify up-selling and cross-selling opportunities with existing customers [Law01].

- Facilitating efficient browsing and searching of the web by hierarchically clustering web-pages.

The challenges in both of these applications mainly arise from two aspects:

1. large numbers of data samples, n, and

2. each sample having *a large number of attributes or features* (dimensions, d).

Certain data mining applications have the additional challenge of how to deal with seasonality and other temporal variations in the data. This aspect is not within the scope of this dissertation, but see [GG01].

The first aspect is typically dealt with by subsampling the data, exploiting summary statistics, aggregating or 'rolling up' to consider data at a

coarser resolution, or by using approximating heuristics that reduce computation time at the cost of some loss in quality. See [HKT01], chapter 8 for several examples of such approaches.

The second aspect is typically addressed by reducing the number of features, by either selection of a subset based on a suitable criteria, or by transforming the original set of attributes into a smaller one using linear projections (e.g., Principal Component Analysis (PCA)) or through non-linear [CG01] means. Extensive approaches for feature selection or extraction have been long studied, particularly in the pattern recognition community [YC74, MJ95, DHS01]. If these techniques succeed in reducing the number of (derived) features to the order of 10 or less without much loss of information, then a variety of clustering and visualization methods can be applied to this reduced dimensionality feature space. Otherwise, the problem may still be tractable if the data is faithful to certain simplifying assumptions, most notably, that either (i) the features are class- or cluster-conditionally independent, or that (ii) most of the data can be accounted for by a 2- or 3-dimensional manifold within the high-dimensional embedding space. The simplest example of case (i) is where the data is well characterized by the superposition of a *small* number of Gaussian components with identical *and* isotropic covariances, in which case k-means can be directly applied to a high-dimensional feature space with good results. If the components have different covariance matrices that are still diagonal (or else the number of parameters will grow quadratically), unsupervised Bayes or mixture-density modeling with EM, can be fruitfully applied. For situation (ii), nonlinear PCA, Self-Organizing Map (SOM), Multi-Dimensional Scaling (MDS) or more efficient custom formulations such as FASTMAP [FL95], can be effectively applied.

This chapter primarily addresses the second aspect by describing an alternate way of clustering and visualization when, *even after feature reduction, one is left with hundreds of dimensions per object* (and further reduction will significantly degrade the results), and moreover, simplifying data modeling assumptions are also not valid. In such situations, one is truly faced with the 'curse of dimensionality' issue [Fri94]. We have repeatedly encountered such situations when examining retail industry market-basket data for behavioral customer clustering, and also certain web-based data collections.

Since clustering basically involves grouping objects based on their inter-relationships or similarities, one can alternatively work in *similarity space* instead of the original feature space. The key insight in this work is that if one can find a similarity measure (derived from the object features) that is appropriate for the problem domain, then a single number can capture the essential 'closeness' of a given pair of objects, and any further analysis can be based only on these numbers. The similarity space also lends itself to a simple technique to visualize the clustering results. A major contribution of this chapter is to demonstrate that this technique has increased power when the clustering method used contains ordering information (e.g., *top-down*). Popular clustering methods in feature space are either non-hierarchical (as in k-means), or bottom-up (agglomerative clustering). However, if one transforms the clustering problem into a related problem of partitioning a similarity graph, several powerful partitioning methods with ordering properties (as described in the introductory paragraph) can be applied. Moreover, the overall framework is quite generally applicable if one can determine the appropriate similarity measure for a given situation. This chapter applies it to three different domains (i) clustering market-baskets (ii) web-documents and (iii) web-logs. In each

situation, a suitable similarity measure emerges from the domain's specific needs.

The overall technique for clustering and visualization is linear in the number of dimensions, but with respect to the number of data points, n, it is quadratic in computational and storage complexity. This can become problematic for very large databases. Several methods for reducing this complexity are outlined in section 3.6, but not elaborated upon much as that is not the primary focus of this present work.

To make concrete some of the remarks above and motivate the rest of the chapter, let us take a closer look at transactional data. A large market-basket database may involve thousands of customers and product-lines. Each record corresponds to a store visit by a customer, so that customer could have multiple entries over time. The transactional database can be conceptually viewed as a sparse representation of a product (feature) by customer (object) matrix. The (i, j)-th entry is non-zero only if customer j bought product i in that transaction. In that case, the entry represents pertinent information such as quantity bought or extended price (quantity $\times$ price) paid.

Since most customers only buy a small subset of these products during any given visit, the corresponding feature vector (column) describing such a transaction is (i) High-dimensional (large number of products), but (ii) Sparse (most features are zero). (iii) Also, transactional data typically has significant outliers, such as a few, big corporate customers that appear in an otherwise small retail customer data. Filtering these outliers may not be easy, nor desirable since they could be very important (e.g., major revenue contributors). (iv) In addition, features are often neither nominal nor continuous, but have discrete positive ordinal attribute values, with a strongly non-Gaussian distri-

bution.

One way to reduce the feature space is to consider only the most dominant products (attribute selection), but in practice this may still leave hundreds of products to be considered. And since product popularity tends to follow a Zipf distribution [Zip29], the tail is 'heavy', meaning that *revenue* contribution from the less popular products is significant for certain customers. Moreover, in retail the higher *profit margins* are often associated with less popular products. One can do a 'roll-up' to reduce number of products, but with a corresponding loss in resolution or granularity. Feature extraction or transformation is typically not carried out as derived features lose the semantics of the original ones as well as the sparsity property.

The alternative to attribute reduction is to try '*simplification via modeling*'. One approach would be to only consider binary features (bought or not). This reduces each transaction to an unordered set of the purchased products. Thus one can use techniques such as the a-priori algorithm to determine associations or rules. In fact, this is currently the most popular approach to market-basket analysis (see [BL97], chapter 8). Unfortunately, this results in loss of vital information: one cannot differentiate between buying 1 gallon of milk and 100 gallons of milk, or one cannot weight importance between buying an apple vs. buying a car, although clearly these are very different situations from a business perspective. In general, association-based rules derived from such sets will be inferior when revenue or profits are the primary performance indicators, since the simplified data representation loses information about quantity, price or margins. The other broad class of modeling simplifications for market-basket analysis is based on taking a macro-level view of the data having characteristics capturable in a small number of parameters. In retail,

a 5-dimensional model for customers composed from indicators for Recency, Frequency, Monetary value, Variety, and Tenure (RFMVT) is popular. However, this useful model is at a much lower resolution that looking at individual products and fails to capture actual purchasing behavior in more complex ways such as taste / brand preferences, or price sensitivity,

Due to all the above issues, traditional vector-space-based clustering techniques work poorly on real-life market-basket data. For example, a typical result of hierarchical agglomerative clustering (both single-link and complete link approaches) on market-basket data is to obtain one huge cluster near the origin, since most customers buy very few items[2], and a few scattered clusters otherwise. Applying k-means could forcibly split this huge cluster into segments depending on the initialization, but not in a meaningful manner. In contrast, the similarity-based methods for both clustering and visualization proposed in this chapter yield far better results for such transactional data. While the methods have certain properties tailored to such data-sets, they can also be applied to other higher dimensional data-sets with similar characteristics. This is illustrated by results on clustering text documents, each characterized by a bag-of-words and represented by a vector of (suitably normalized) term occurrences, often 1000 or more in length. Our detailed comparative study in chapter 4 will show that in this domain traditional clustering techniques also have some difficulties, though not as much as for market-basket data since simplifying assumptions regarding class or cluster conditional independence of features are not violated as much, and consequently both Naive Bayes [MN98] and a normalized version of k-means [DM01] also show decent results. We also apply the technique to clustering visitors to a website based

[2]This is the dilution effect described in [GRS99].

on their footprints, where, once a domain specific suitable similarity metric is determined, the general technique again provides nice results.

We begin by considering domain-specific transformations into similarity space in section 3.2. Section 3.3 describes a specific clustering technique (OPOSSUM), based on a multi-level graph partitioning algorithm [KK98a]. In section 3.4, we describe a simple but effective visualization technique that is applicable to similarity spaces (CLUSION). Clustering and visualization results are presented in section 3.5. In section 3.6, we consider system issues and briefly discuss several strategies to scale OPOSSUM for large data-sets.

3.2 Domain Specific Features and Similarity Space

Notation. Let n be the number of objects / samples / points (e.g., customers, documents, web-sessions) in the data and d the number of features (e.g., products, words, web-pages) for each sample $\mathbf{x}_j$ with $j \in \{1, \ldots, n\}$. Let k be the desired number of clusters. The input data can be represented by a $d \times n$ data matrix $\mathbf{X}$ with the j-th column vector representing the sample $\mathbf{x}_j$. $\mathbf{x}_j^\dagger$ denotes the transpose of $\mathbf{x}_j$. Hard clustering assigns a label $\lambda_j \in \{1, \ldots, k\}$ to each d-dimensional sample $\mathbf{x}_j$, such that similar samples get the same label. In general the labels are treated as nominals with no inherent order, though in some cases, such as 1-dimensional SOMs, any top-down recursive bisection approach as well as our proposed method, the labeling contains extra ordering information. Let $\mathcal{C}_\ell$ denote the set of all objects in the ℓ-th cluster ($\ell \in \{1, \ldots, k\}$), with $\mathbf{x}_j \in \mathcal{C}_\ell \Leftrightarrow \lambda_j = \ell$ and $n_\ell = |\mathcal{C}_\ell|$.

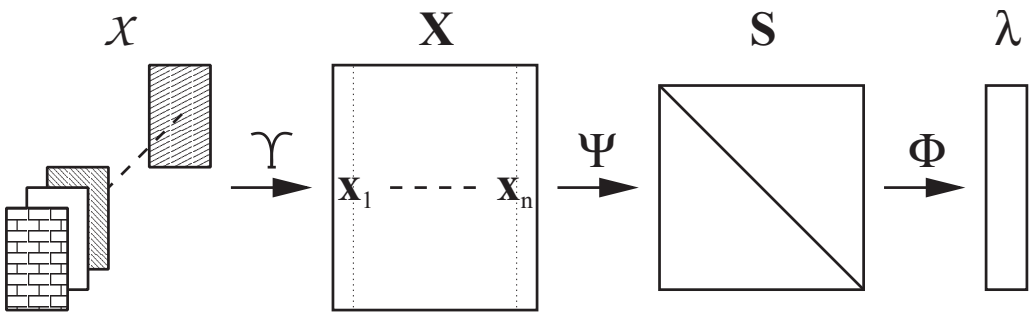


Figure 3.1: The relationship-based clustering framework.

Figure 3.1 gives an overview of our relationship-based clustering **process** from a set of raw object descriptions $\mathcal{X}$ (residing in input space $\mathcal{I}$) via the vector space description $\mathbf{X}$ (in feature space $\mathcal{F}$) and relationship description $\mathbf{S}$ (in similarity space $\mathcal{S}$) to the cluster labels λ (in output space $\mathcal{O}$): $(\mathcal{X} \in \mathcal{I}^n) \xrightarrow{\Upsilon} (\mathbf{X} \in \mathcal{F}^n \subset \mathbb{R}^{d \times n}) \xrightarrow{\Psi} (\mathbf{S} \in \mathcal{S}^{n \times n} = [0,1]^{n \times n} \subset \mathbb{R}^{n \times n}) \xrightarrow{\Phi} (\lambda \in \mathcal{O}^n = \{1, \ldots, k\}^n)$. For example in web-page clustering, $\mathcal{X}$ is a collection of n web-pages x_j with $j \in \{1, \ldots, n\}$. Extracting features using Υ yields $\mathbf{X}$, the term frequencies of stemmed words, normalized such that for all documents $\mathbf{x} : \|\mathbf{x}\|_2 = 1$. Similarities are computed, using e.g., cosine-based similarity Ψ yielding the $n \times n$ similarity matrix $\mathbf{S}$. Finally, the cluster label vector λ is computed using a clustering function Φ, such as graph partitioning. In short, the basic process can be denoted as $\mathcal{X} \xrightarrow{\Upsilon} \mathbf{X} \xrightarrow{\Psi} \mathbf{S} \xrightarrow{\Phi} \lambda$.

Similarity Measures. In this dissertation, we prefer working in similarity space rather than the original vector space in which the feature vectors reside. A similarity measure captures the relationship between two d-dimensional objects in a single number (using on the order of the number of non-zero entries in the vectors or d, at worst, computations). Once this is

done, the original high-dimensional space is not dealt with at all, we only work in the transformed similarity space, and subsequent processing is independent of d.

A similarity measure $\in [0, 1]$ captures how related two data-points $\mathbf{x}_a$ and $\mathbf{x}_b$ are. It should be symmetric ($s(\mathbf{x}_a, \mathbf{x}_b) = s(\mathbf{x}_b, \mathbf{x}_a)$), with self-similarity $s(\mathbf{x}_a, \mathbf{x}_a) = 1$. However, in general, similarity functions (respectively their distance function equivalents $\delta = \sqrt{-\log(s)}$, see below) do *not* obey the triangle inequality.

An obvious way to compute similarity is through a suitable monotonic and inverse function of a Minkowski (L_p) distance, δ. Candidates include $s = 1/(1 + \delta)$ and $s = e^{-\delta^2}$, the later being preferable due to maximum likelihood properties (see chapter 4). Similarity can also be defined by the cosine of the angle between two vectors:

$$s^{(\mathrm{C})}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^\dagger \mathbf{x}_b}{\|\mathbf{x}_a\|_2 \cdot \|\mathbf{x}_b\|_2} \tag{3.1}$$

Cosine similarity is widely used in text clustering because two documents with the same proportions of term occurrences but different lengths are often considered identical. In retail data such normalization loses important information about the life-time customer value, and we have recently shown that the **extended Jaccard similarity** measure is more appropriate [SG00c]. For binary features, the Jaccard coefficient [JD88] (also known as the Tanimoto coefficient [DHS01]) measures the ratio of the intersection of the product sets to the union of the product sets corresponding to transactions $\mathbf{x}_a$ and $\mathbf{x}_b$, each having binary (0/1) elements.

$$s^{(\mathrm{J})}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^\dagger \mathbf{x}_b}{\|\mathbf{x}_a\|_2^2 + \|\mathbf{x}_b\|_2^2 - \mathbf{x}_a^\dagger \mathbf{x}_b} \tag{3.2}$$

The extended Jaccard coefficient is also given by equation 3.2, but allows elements of $\mathbf{x}_a$ and $\mathbf{x}_b$ to be arbitrary positive real numbers. This coefficient captures a vector-length-sensitive measure of similarity. However, it is still invariant to scale (dilating $\mathbf{x}_a$ and $\mathbf{x}_b$ by the same factor does not change $s(\mathbf{x}_a, \mathbf{x}_b)$). A detailed discussion of the properties of various similarity measures can be found in chapter 4.

Since, for general data distributions, one cannot avoid the 'curse of dimensionality', there is no similarity metric that is optimal for all applications. Rather, one needs to to determine an appropriate measure for the given application, that captures the essential aspects of the class of high-dimensional data distributions being considered.

3.3 OPOSSUM

In this section, we propose OPOSSUM (Optimal Partitioning of Sparse Similarities Using Metis), a similarity-based clustering technique particularly tailored to market-basket data. OPOSSUM differs from other graph-based clustering techniques by application-driven balancing of clusters, non-metric similarity measures, and visualization driven heuristics for finding an appropriate k.

3.3.1 Balancing

Typically, one segments transactional data into 7-14 groups, each of which should be of comparable importance. Balancing avoids trivial clusterings (e.g., $k-1$ singletons and 1 big cluster). More importantly, the desired balancing properties have many application driven advantages. For example when each

cluster contains the same number of customers, discovered phenomena (e.g. frequent products, co-purchases) have equal significance / support and are thus easier to evaluate. When each customer cluster equals the same revenue share, marketing can spend an equal amount of attention and budget to each of the groups. OPOSSUM strives to deliver 'balanced' clusters using either of the following two criteria:

- *Sample balanced:* Each cluster should contain roughly the same number of samples, n/k. This allows, for example, retail marketers to obtain a customer segmentation with equally sized customer groups.

- *Value balanced:* Each cluster should contain roughly the same amount of feature values. Thus, a cluster represents a k-th fraction of the total feature value $v = \sum_{j=1}^{n} \sum_{i=1}^{d} x_{i,j}$. In customer clustering, we use extended price per product as features and, thus, each cluster represents a roughly equal contribution to total revenue. In web-session clustering the feature of choice is the time spent on a particular web-page. This results in user clusters balanced with respect to the total time spent on the site.

We formulate the desired balancing properties by assigning each object (customer, document, web-session) a weight and then softly constrain the sum of weights in each cluster. For sample balanced clustering, we assign each sample $\mathbf{x}_j$ the same weight $w_j = 1/n$. To obtain value balancing properties, a sample $\mathbf{x}_j$'s weight is set to $w_j = \frac{1}{v} \sum_{i=1}^{d} x_{i,j}$. Please note that the sum of weights for all samples is 1.

3.3.2 Vertex Weighted Graph Partitioning

We map the problem of clustering to partitioning a vertex weighted graph $\mathcal{G}$ into k unconnected components of approximately equal size (as defined by the balancing constraint) by removing a minimal amount of edges. The objects to be clustered are viewed as a set of vertices $\mathcal{V} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Two vertices $\mathbf{x}_a$ and $\mathbf{x}_b$ are connected with an undirected edge $(a, b) \in \mathcal{E}$ of positive weight given by the similarity $s(\mathbf{x}_a, \mathbf{x}_b)$. This defines the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. An edge separator $\Delta\mathcal{E}$ is a set of edges whose removal splits the graph $\mathcal{G}$ into k pair-wise unconnected components (sub-graphs) $\{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$. All sub-graphs $\mathcal{G}_\ell = (\mathcal{V}_\ell, \mathcal{E}_\ell)$ have pairwise disjoint sets of vertices and edges. The edge separator for a particular partitioning includes all the edges that are not part of any sub-graph, or $\Delta\mathcal{E} = (\mathcal{E} \setminus (\mathcal{E}_1 \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_k))$. The clustering task is thus to find an edge separator with a minimum sum of edge weights, that partitions the graph into k disjoint pieces. The following equation formalizes this *minimum cut objective*:

$$\min_{\Delta\mathcal{E}} \sum_{(a,b)\in\Delta\mathcal{E}} s(\mathbf{x}_a, \mathbf{x}_b) \tag{3.3}$$

Without loss of generality, we can assume that the vertex weights w_j are normalized to sum up to 1: $\sum_{j=1}^{n} w_j = 1$. While striving for the minimum cut objective, *the balancing constraint*

$$k \cdot \max_{\ell\in\{1,\ldots,k\}} \sum_{\lambda_j=\ell} w_j \leq t \tag{3.4}$$

has to be fulfilled. The left hand side of the inequality is called the imbalance (the ratio of the biggest cluster in terms of cumulative normalized edge weight to the desired equal cluster-size $1/k$) and has a lower bound of 1. The balancing

threshold t enforces perfectly balanced clusters for $t = 1$. In practice t is often chosen to be slightly greater than 1 (e.g., we use $t = 1.05$ for all our experiments which allows at most 5% of imbalance).

Thus, in graph partitioning one has to essentially solve a constrained optimization problem. Finding such an optimal partitioning is an NP-hard problem [GJ79]. However, there are fast, heuristic algorithms for this widely studied problem. We experimented with the Kernighan-Lin (KL) algorithm, recursive spectral bisection, and multi-level k-way partitioning (METIS).

The basic idea in KL [KL70] to dealing with graph partitioning is to construct an initial partition of the vertices either randomly or according to some problem-specific strategy. Then the algorithm sweeps through the vertices, deciding whether the size of the cut would increase or decrease if we moved this vertex $\mathbf{x}$ over to another partition. The decision to move $\mathbf{x}$ can be made in time proportional to its degree by simply counting whether more of $\mathbf{x}$'s neighbors are on the same partition as $\mathbf{x}$ or not. Of course, the desirable side for $\mathbf{x}$ will change if many of its neighbors switch, so multiple passes are likely to be needed before the process converges to a local optimum.

In recursive bisection, a k-way split is obtained by recursively partitioning the graph into two subgraphs. Spectral bisection [PSL90, HL95] uses the eigenvector associated with the second smallest eigenvalue of the graph's Laplacian (Fiedler vector) [Fie75] for splitting.

METIS [KK98a] handles multi-constraint multi-objective graph partitioning in three phases: (i) coarsening, (ii) initial partitioning, and (iii) refining. First a sequence of successively smaller and therefore coarser graphs is constructed through heavy-edge matching. Secondly, the initial partitioning is constructed using one out of four heuristic algorithms (three based on graph

growing and one based on spectral bisection). In the third phase the coarsened partitioned graph undergoes boundary Kernighan-Lin refinement. In this last phase vertices are only swapped amongst neighboring partitions (boundaries). This ensures that neighboring clusters are more related than non-neighboring clusters. This ordering property is beneficial for visualization, as explained in subsection 3.6.1. In contrast, since recursive bisection computes a bisection of a subgraph at a time, its view is limited. Thus, it can not fully optimize the partition ordering and the global constraints. This renders it less effective for our purposes. Also, we found the multi-level partitioning to deliver the best partitionings as well as to be the fastest and most scalable of the three choices we investigated. Hence, METIS is used as the graph partitioner in OPOSSUM.

3.3.3 Determining the Number of Clusters

Finding the 'right' number of clusters k for a data-set is a difficult and often ill-posed problem, since even for the same data-set, there can be several answers depending on the scale or granularity one is interested in. In probabilistic approaches to clustering, likelihood-ratios, Bayesian techniques and Monte Carlo cross-validation are popular. In non-probabilistic methods, a regularization approach, which penalizes for large k, is often adopted. If the data is labelled, then mutual information between cluster and class labels can be used to determine the number of clusters. Other metrics such as purity of clusters or entropy are of less use as they are biased towards a larger number of clusters (see chapter 4).

For transactional data, often the number is specified by the end-user

to be typically between 7 and 14 [BL97]. Otherwise, one can employ a suitable heuristic to obtain an appropriate value of k during the clustering process. This subsection describes how we find a desirable clustering, with *high overall cluster quality* $\phi^{(\mathrm{Q})}$ and a *small number of clusters* k. Our objective is to maximize intra-cluster similarity and minimize inter-cluster similarity, given by $\mathrm{intra}(\mathbf{X}, \lambda, i) = \frac{2}{(n_i-1)\cdot n_i} \sum_{\lambda_a=\lambda_b=i, b>a} s(\mathbf{x}_a, \mathbf{x}_b)$ and $\mathrm{inter}(\mathbf{X}, \lambda, i, j) = \frac{1}{n_i \cdot n_j} \sum_{\lambda_a=i, \lambda_b=j} s(\mathbf{x}_a, \mathbf{x}_b)$, respectively, where i and j are cluster indices. Note that intra-cluster similarity is undefined (0/0) for singleton clusters. Hence, we define our *quality* measure $\phi^{(\mathrm{Q})} \in [0, 1]$ ($\phi^{(\mathrm{Q})} < 0$ in case of pathological / inverse clustering) based on the ratio of weighted average inter-cluster to weighted average intra-cluster similarity:

$$\phi^{(\mathrm{Q})}(\mathbf{X}, \lambda) = 1 - \frac{\sum_{i=1}^{k} \frac{n_i}{n-n_i} \sum_{j \in \{1,\ldots,i-1,i+1,\ldots,k\}} n_j \cdot \mathrm{inter}(\mathbf{X}, \lambda, i, j)}{\sum_{i=1}^{k} n_i \cdot \mathrm{intra}(\mathbf{X}, \lambda, i)} \tag{3.5}$$

$\phi^{(\mathrm{Q})} = 0$ indicates that samples within the same cluster are on average not more similar than samples from different clusters. On the contrary, $\phi^{(\mathrm{Q})} = 1$ describes a clustering where every pair of samples from different clusters has the similarity of 0 and at least one sample pair from the same cluster has a non-zero similarity. Note that our definition of quality does not take the 'amount of balance' into account, since balancing is already observed fairly strictly by the constraints in the graph partitioning step.

To achieve a high quality $\phi^{(\mathrm{Q})}$ as well as a low k, the target function $\phi^{(\mathrm{T})} \in [0, 1]$ is the product of the quality $\phi^{(\mathrm{Q})}$ and a penalty term which works very well in practice. If $n \geq 4$ and $2 \leq k \leq \lfloor n/2 \rfloor$, then there exists at least one clustering with no singleton clusters. The penalized quality gives the penalized quality $\phi^{(\mathrm{T})}$ and is defined as $\phi^{(\mathrm{T})}(k) = \left(1 - \frac{2k}{n}\right) \cdot \phi^{(\mathrm{Q})}(k)$. A modest linear penalty was chosen, since our quality criterion does not necessarily improve

with increasing k (unlike e.g. the squared error criterion). For large n, we search for the optimal k in the entire window from $2 \leq k \leq 100$. In many cases, however, a forward search starting at $k = 2$ and stopping at the first down-tick of penalized quality while increasing k is sufficient.

Finally, a practical alternative, as exemplified by the experimental results later, is to first over-cluster and then use the visualization aid to combine clusters as needed (subsection 3.5.2).

3.4 CLUSION: Cluster Visualization

In this section, we present our visualization tool, highlight some of its properties and compare it with some popular visualization methods. Applications of this tool are illustrated in section 3.5.

3.4.1 Coarse Seriation

When data is limited to 2 or 3 dimensions, the most powerful tool for judging cluster quality is usually the human eye. CLUSION, our CLUSter visualizatION toolkit, allows us to convert high-dimensional data into a perceptually more suitable format, and employ the human vision system to explore the *relationships* in the data, *guide* the clustering process, and *verify* the quality of the results. In our experience with two years of Dell customer data, we found CLUSION effective for getting clusters balanced w.r.t. number of customers or net dollar ($) amount, and even more so for conveying the results to marketing management.

CLUSION looks at the output of a clustering routine, reorders the data points such that points with the same cluster label are contiguous, and then

visualizes the resulting permuted similarity matrix, $\mathbf{S}'$. More formally, the original $n \times n$ similarity matrix $\mathbf{S}$ is permuted with a $n \times n$ permutation matrix $\mathbf{P}$ which is defined as follows:

$$p_{i,j} = \begin{cases} 1 \text{ if } j = \sum_{a=1}^{i} l_{a,\lambda_i} + \sum_{\ell=1}^{\lambda_i - 1} n_\ell \\ 0 \text{ otherwise} \end{cases} \tag{3.6}$$

l are entries in the binary $n \times k$ cluster membership indicator matrix $\mathbf{L}$:

$$l_{i,j} = \begin{cases} 1 \text{ if } \lambda_i = j \\ 0 \text{ otherwise} \end{cases} \tag{3.7}$$

In other words, $p_{i,j}$ is 1 if j is the sum of the number of points amongst the first i that belong to the same cluster and the number of points in the first $\lambda_i - 1$ clusters. Now, the permuted similarity matrix $\mathbf{S}'$ and the corresponding label vector λ' and data matrix $\mathbf{X}'$ are:

$$\mathbf{S}' = \mathbf{P}\mathbf{S}\mathbf{P}^\dagger \quad , \quad \lambda' = \mathbf{P}\lambda \quad , \quad \mathbf{X}' = \mathbf{P}\mathbf{X} \tag{3.8}$$

For a 'good' clustering algorithm and $k \to n$ this is related to sparse matrix reordering, for this results in the generation of a 'banded matrix' where high entries should all fall near the diagonal line from the upper left to the lower right of the matrix. Since equation 3.8 is essentially a partial ordering operation we also refer to it as coarse *seriation*, a phrase used in disciplines such as anthropology and archaeology to describe the reordering of the primary data matrix so that similar structures (e.g., genetic sequences) are brought closer [Mur85, ESBB98].

3.4.2 Visualization

The seriation of the similarity matrix, $\mathbf{S}'$, is very useful for visualization. Since the similarity matrix is 2-dimensional, it can be readily visualized as a gray-

level image where a white (black) pixel corresponds to minimal (maximal) similarity of 0 (1). The darkness (gray level value) of the pixel at row a and column b increases with the similarity between the samples $\mathbf{x}_a$ and $\mathbf{x}_b$. When looking at the image it is useful to consider the similarity s as a random variable taking values from 0 to 1. The expected similarity *within* cluster ℓ is thus represented by the average intensity within a square region with side length n_ℓ, around the main diagonal of the matrix. The off-diagonal rectangular areas visualize the relationships *between* clusters. The brightness distribution in the rectangular areas yields insight towards the quality of the clustering and possible improvements. In order to make these regions apparent, thin red horizontal and vertical lines are used to show the divisions into the rectangular regions[3]. Visualizing similarity space in this way can help to quickly get a feel for the clusters in the data. Even for a large number of points, a sense for the intrinsic number of clusters k in a data-set can be gained.

Figure 3.2 shows CLUSION output in four extreme scenarios to provide a feel for how data properties translate to the visual display. Without any loss of generality, we consider the partitioning of a set of objects into 2 clusters. For each scenario, on the left hand side the original similarity matrix $\mathbf{S}$ and the seriated version $\mathbf{S}'$ (CLUSION) for an optimal bipartitioning is shown. On the right hand side four histograms for the distribution of similarity values s, which range from 0 to 1, are shown. From left to right, we have plotted: distribution of s over the entire data, within the first cluster, within the second cluster, and between first and second cluster. If the data is naturally clustered and the clustering algorithm is good, then the middle two columns of plots will be much more skewed to the right as compared to the first and fourth

[3]This can be more clearly seen in the color pictures in the soft-copy.

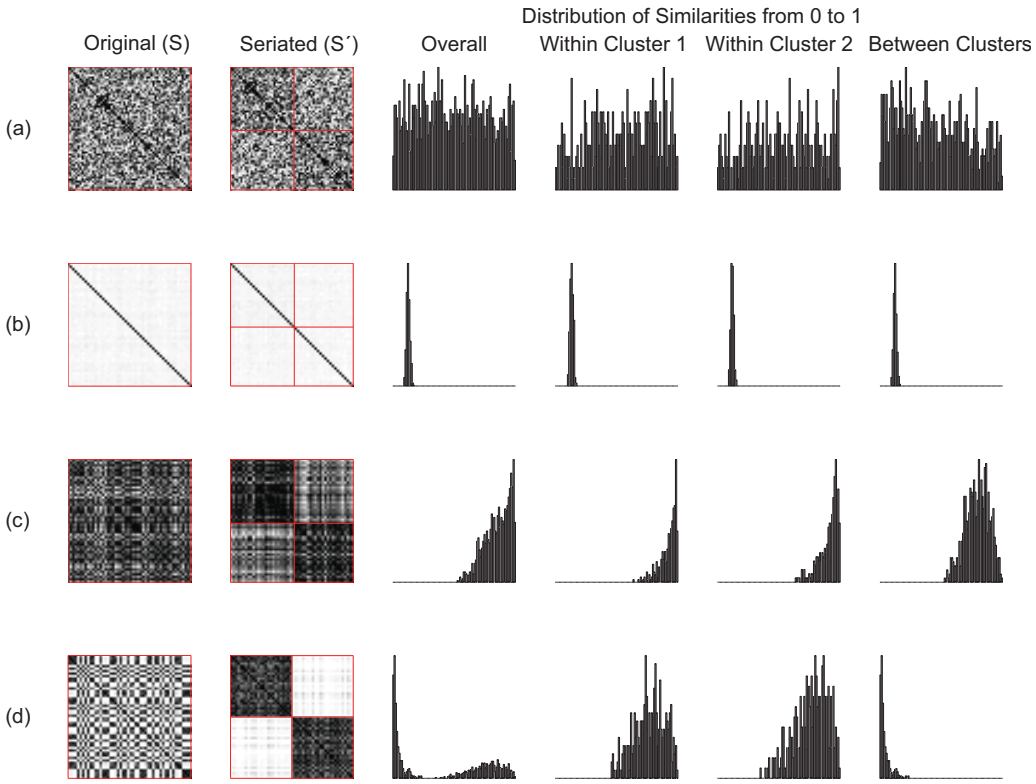


Figure 3.2: Illustrative CLUSION patterns in original order and seriated using optimal bipartitioning are shown in the left two columns. The right four columns show corresponding similarity distributions. In each example there are 50 objects: (a) no natural clusters (randomly related objects), (b) set of singletons (pairwise near orthogonal objects), (c) one natural cluster (unimodal Gaussian), (d) two natural clusters (mixture of two Gaussians)

columns. In our visualization this corresponds to brighter off-diagonal regions and darker block diagonal regions in $\mathbf{S}'$ as compared to the original $\mathbf{S}$ matrix.

The proposed visualization technique is quite powerful and versatile. In figure 3.2(a) the chosen similarity behaves randomly. Consequently, no strong visual difference between on- and off-diagonal regions can be perceived with CLUSION in $\mathbf{S}'$. It indicates clustering is ineffective which is expected since there is no structure in the similarity matrix. Figure 3.2(b) is based on data consisting of pair-wise almost equi-distant singletons. Clustering into two groups still renders the on-diagonal regions very bright suggesting more splits. In fact, this will remain unchanged until each data-point is a cluster by itself, thus, revealing the singleton character of the data. For monolithic data (figure 3.2(c)), many strong similarities are indicated by an almost uniformly dark similarity matrix $\mathbf{S}$. Splitting the data results in dark off-diagonal regions in $\mathbf{S}'$. A dark off-diagonal region suggests that the clusters in the corresponding rows and columns should be merged (or not be split in the first place). CLUSION indicates that this data is actually one large cluster. In figure 3.2(d), the gray-level distribution of $\mathbf{S}$ exposes bright as well as dark pixels, thereby recommending it should be split. In this case, $k = 2$ apparently is a very good choice (and the clustering algorithm worked well) because in $\mathbf{S}'$ on-diagonal regions are uniformly dark and off-diagonal regions are uniformly bright.

This induces an intuitive mining process that guides the user to the 'right' number of clusters. Too small a k leaves the on-diagonal regions inhomogeneous. On the contrary, growing k beyond the natural number of clusters will introduce dark off-diagonal regions. Finally, CLUSION can be used to visually compare the appropriateness of different similarity measures. Let us assume, for example, that each row in figure 3.2 illustrates a particular way of

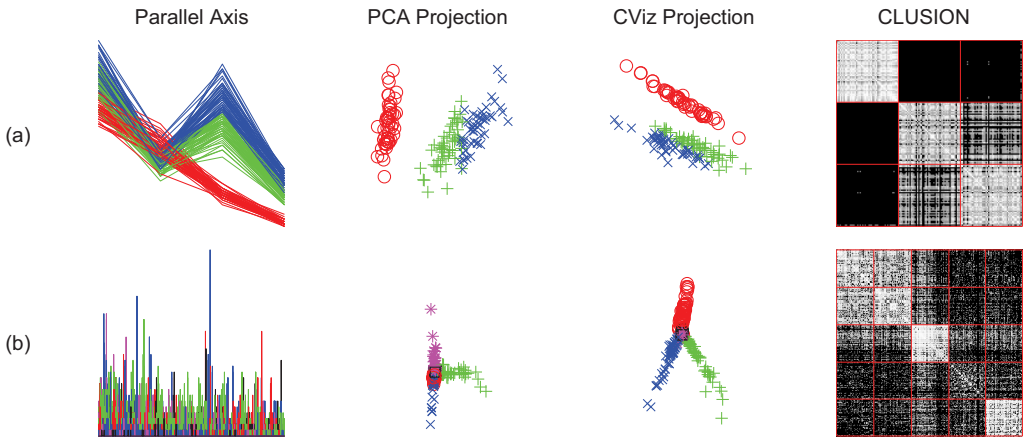


Figure 3.3: Comparison of cluster visualization techniques. All tools work well on the 4-dimensional `IRIS` data (a). But on the 2903-dimensional `YAHOO` news document data (b), only CLUSION reveals that clusters 1 and 2 are actually highly related, cluster 3 is strong and interdisciplinary, 4 is weak, and 5 is strong.

defining similarity for the same data-set. Then, CLUSION makes visually apparent that the similarity measure in (d) lends itself much better for clustering than the measures illustrated in rows (a), (b), and (c).

3.4.3 Comparison

CLUSION gives a *relationship-centered* view, as contrasted with common projective techniques, such as the selection of dominant features or optimal linear projections (PCA), which are *object-centered*. In CLUSION, the actual features are transparent, instead, all pair-wise relationships, the relevant aspect for the purpose of clustering, are displayed.

Figure 3.3 compares CLUSION with some other popular visualizations.

In figure 3.3(a) parallel axis, PCA projection, CViz (projection through plane defined by centroids of clusters 1, 2, and 3) as well as CLUSION succeed in visualizing the `IRIS` data (see also appendix A.2). Membership in cluster 1 / 2 / 3 is indicated by colors red / blue / green (parallel axis), colors red / blue / green and shapes $\circ/\times/+$ (PCA and CViz), and position on diagonal from upper left to lower right corner (CLUSION), respectively. All four tools succeed in visualizing three clusters and making apparent that clusters 2 and 3 are closer than any other and cluster 1 is very compact.

Figure 3.3(b) shows the same comparison for 293 documents from which 2903 word frequencies where extracted to be used as features. In fact this data consists of 5 clusters selected from 40 clusters extracted from a Yahoo! news document collection which will be described in more detail in subsection 3.5.2 (`YAHOO`). The colors black / magenta and the shapes $\square$ / $*$ have been added to indicate cluster 4 / 5, respectively. The parallel axis plot becomes useless clutter due to the high number of dimensions as well as the large number of objects. PCA and CViz succeed in separating three clusters each (2, 3, 5 and 1, 2, 3, respectively) and show all others superimposed on the axis origin. They give no suggestions towards which clusters are compact or which clusters are related. Only CLUSION suggests that clusters 1 and 2 are actually highly related, cluster 3 is interdisciplinary, 4 is weak, and 5 is a strong cluster. And indeed, when looking at the cluster descriptions (which might not be so easily available and understandable in all domains), the intuitive interpretations revealed by CLUSION are proven to be very true:

cluster	dominant category	purity	entropy	most frequent word stems
1	`health` (H)	100%	0.00	hiv, depress, immun
2	`health` (H)	100%	0.00	weight, infant, babi
3	`online` (o)	58%	0.43	apple, intel, electron
4	`film` (f)	38%	0.72	hbo, ali, alan
5	`television` (t)	83%	0.26	household, sitcom, timeslot

Note that the majority category, purity, and entropy are only available where a supervised categorization is given. Of course the categorization cannot be used to tune the clustering. Clusters 1 and 2 contains only documents from the `Health` category so they are highly related. The 4th cluster, which is indicated to be weak by CLUSION, has in fact the lowest purity in the group with 38% of documents from the most dominant category (`film`). CLUSION also suggests cluster 3 is not only strong, as indicated by the dark diagonal region, but also has distinctly above average relationships to *all* other 4 clusters. On inspecting the word stems typifying this cluster (Apple, Intel, and electron(ics)) it is apparent that this is because of the interdisciplinary appearance of technology savvy words in recent news releases. Since such cluster descriptions might not be so easily available or well understood in all domains, the intuitive display of CLUSION is very useful.

CLUSION has several other powerful properties. For example, it can be integrated with product hierarchies (meta-data) to provide simultaneous customer and product clustering, as well as multi-level views / summaries. It also has a graphical user interface so one can interactively browse / split / merge a data-set which is of great help to speed-up the iterations of analysis

during a data mining project.

3.5 Experiments

3.5.1 Retail Market-basket Clusters

First, we will show clusters in a real retail transaction database of 21672 customers of a drugstore[4]. For the purpose of the experiments in this chapter, we randomly selected 2500 customers. The total number of transactions (cash register scans) for these customers is 33814 over a time interval of three months. We rolled up the product hierarchy once to obtain 1236 different products purchased. 15% of the total revenue is contributed by the single item `Financial-Depts` (on site financial services such as check cashing and bill payment) which was removed because it was too common. 473 of these products accounted for less than $25 each in toto and were dropped. The remaining $n = 2466$ customers (34 customers had empty baskets after removing the irrelevant products) with their $d = 762$ features compose the `RETAIL` data-set. Appendix A.4 gives exemplary transactions and illustrates Zipf-like distributions found in the data. The customers in `RETAIL` were clustered using OPOSSUM. The extended price was used as the feature entries to represent purchased quantity weighted according to price.

In this customer clustering case study we set $k = 20$. In this application domain, the number of clusters is often predetermined by marketing considerations such as advertising industry standards, marketing budgets, marketers ability to handle multiple groups, and the cost of personalization. In general,

[4]provided by Knowledge Discovery 1

a reasonable value of k can be obtained using heuristics (subsection 3.3.3).

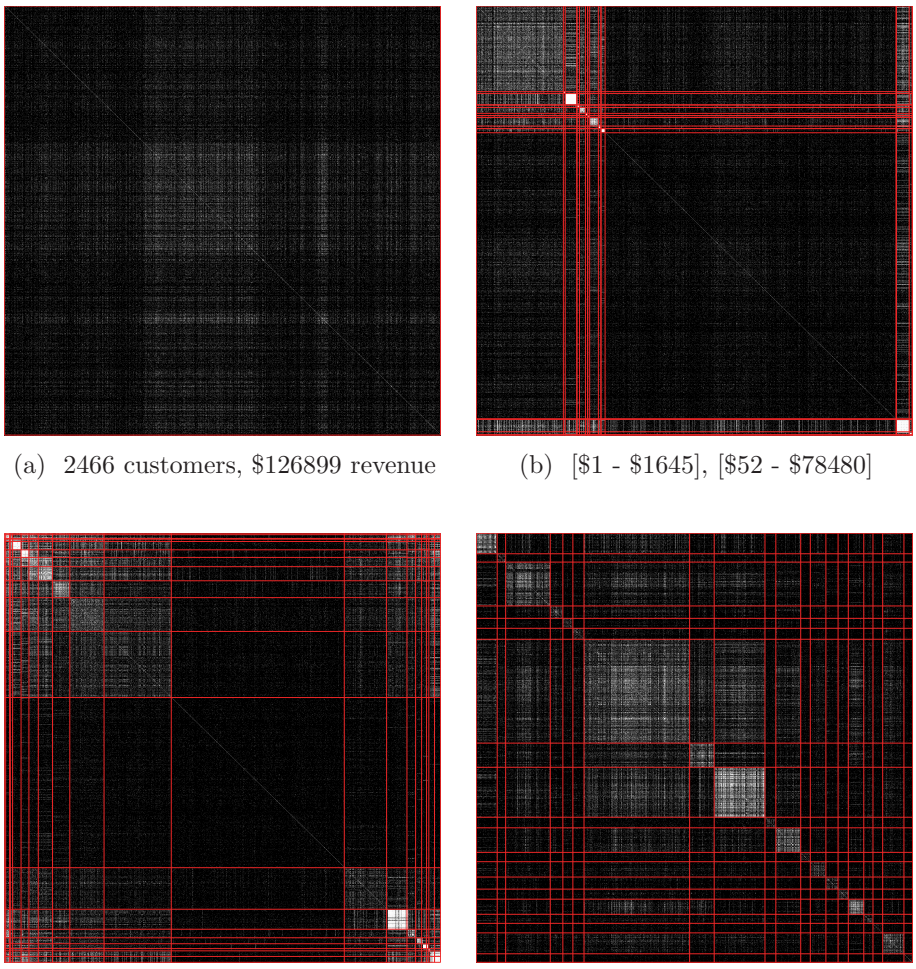
OPOSSUM's results for this example are obtained with a 1.7 GHz Pentium 4 PC with 512 MB RAM in approximately 35 seconds (∼30s file I/O, 2.5s similarity computation, 0.5s conversion to integer weighted graph, 0.5s graph partitioning). Figures 3.4 and 3.5 show the extended Jaccard similarity matrix (83% sparse) using CLUSION in six scenarios: 3.4(a) original (randomly) ordered matrix, 3.4(b) seriated using Euclidean k-means, 3.4(c) using SOM, 3.4(d) using standard Jaccard k-means, 3.5(a) using extended Jaccard sample balanced OPOSSUM, 3.5(b) using value balanced OPOSSUM clustering. Customer and revenue ranges are given below each image. In figure 3.4(a), (b), (c), and (d) clusters are neither compact nor balanced. In figure 3.5(a) and (b) clusters are much more compact, even though there is the additional constraint that they be balanced, based on equal number of customers and equal revenue metrics, respectively. Below each CLUSION visualization, the ranges of numbers of customers and revenue totals in $ amongst the 20 cluster are given to indicate balancedness. We also experimented with minimum distance agglomerative clustering but this resulted in 19 singletons and 1 cluster with 2447 customers so we did not bother including this approach. Clearly, k-means in the original feature space, the standard clustering algorithm, does not perform well at all (figure 3.4(b)). The SOM after 100000 epochs performs slightly better (figure 3.4(c)) but is outperformed by the standard Jaccard k-means (figure 3.4(d)) which is adopted to similarity space by using $\sqrt{-\log(s^{(\mathrm{J})})}$ as distances (see chapter 4). As the relationship-based CLUSION shows, OPOSSUM (figure 3.5(a),(b)) gives more compact (better separation of on- and off-diagonal regions) and well balanced clusters as compared to all other techniques. For example, looking at standard Jaccard k-means, the clusters contain between

48 and 597 customers contributing between \$608 and \$70443 to revenue[5]. Thus the clusters may not be of comparable importance from a marketing standpoint. Moreover clusters are hardly compact: Darkness is only slightly stronger in the on-diagonal regions in figure 3.4(d). All visualizations have been histogram equalized for printing purposes. However, they are still much better observed by browsing interactively on a computer screen.

A very compact and useful way of profiling a cluster is to look at their most *descriptive* and their most *discriminative* features. For market-basket data, this can be done by looking at a cluster's highest revenue products and the most unusual revenue drivers (e.g., products with highest revenue lift). Revenue lift is the ratio of the average spending on a product in a particular cluster to the average spending in the entire data-set.

In table 3.1 the top three descriptive and discriminative products for the customers in the 20 value balanced clusters are shown (see also figure 3.5(b)). Customers in cluster $\mathcal{C}_2$, for example, mostly spent their money on smoking cessation gum (\$10.15 on average). Interestingly, while this is a 35-fold average spending on smoking cessation gum, these customers also spend 35 times more on blood pressure related items, peanuts and snacks. Do these customers lead an unhealthy lifestyle and are eager to change? Cluster $\mathcal{C}_{15}$, which can be seen to be highly compact cluster of Christmas shoppers characterized by greeting card and candy purchases. Note that OPOSSUM had an extra constraint that clusters should be of comparable value. This may force a larger natural cluster to split, as may be the case causing the similar clusters $\mathcal{C}_9$ and $\mathcal{C}_{10}$. Both are Christmas gift shoppers (table 3.1(top)), cluster $\mathcal{C}_9$ are the moderate spenders

[5]The solution for k-means depends on the initial choices for the means. A representative solution is shown here.



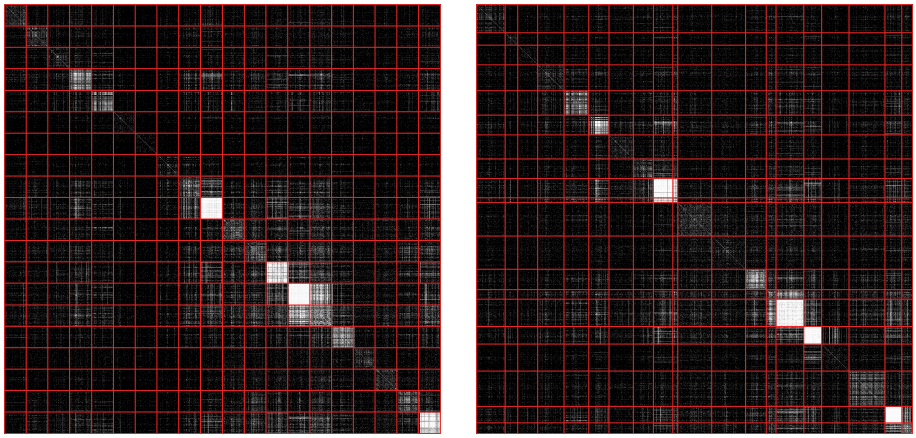
(a) 2466 customers, $126899 revenue

(b) [$1 - $1645], [$52 - $78480]

(c) [4 - 978], [$1261 - $12162]

(d) [48 - 597], [$608 - $70443]

Figure 3.4: Visualizing partitioning drugstore customers from `RETAIL` dataset into 20 clusters. Relationship visualizations using CLUSION: (a) original (randomly) ordered similarity matrix, (b) seriated or partially reordered using Euclidean k-means, (c) using SOM, (d) using standard Jaccard k-means. Customer and revenue ranges are given below each image. See also figure 3.5.



(a) [122 - 125], [$1624 - $14361] (b) [28 - 203], [$6187 - $6609]

Figure 3.5: Visualizing partitioning drugstore customers from `RETAIL` data-set into 20 clusters. Relationship visualizations using CLUSION: (a) using extended Jaccard sample balanced OPOSSUM, (b) using value balanced OPOSSUM clustering. Customer and revenue ranges are given below each image. In figure 3.4(a), (b), (c), and (d) clusters are neither compact nor balanced. In figure 3.5(a) and (b) clusters are much more compact, even thoughthere is the additional constraint that they be balanced, based on equal number of customers and equal revenue metrics, respectively.

$\mathcal{C}_\ell$	top product	$	lift	sec. product	$	lift	third product	$	lift
1	bath gift packs	3.44	7.69	hair growth m	0.90	9.73	boutique island	0.81	2.61
2	smoking cessati	10.15	34.73	tp canning item	2.04	18.74	blood pressure	1.69	34.73
3	vitamins other	3.56	12.57	tp coffee maker	1.46	10.90	underpads hea	1.31	16.52
4	games items 180	3.10	7.32	facial moisturi	1.80	6.04	tp wine jug ite	1.25	8.01
5	batt alkaline i	4.37	7.27	appliances item	3.65	11.99	appliances appl	2.00	9.12
6	christmas light	8.11	12.22	appliances hair	1.61	7.23	tp toaster/oven	0.67	4.03
7	christmas food	3.42	7.35	christmas cards	1.99	6.19	cold bronchial	1.91	12.02
8	girl toys/dolls	4.13	12.51	boy toys items	3.42	8.20	everyday girls	1.85	6.46
9	christmas giftw	12.51	12.99	christmas home	1.24	3.92	christmas food	0.97	2.07
10	christmas giftw	19.94	20.71	christmas light	5.63	8.49	pers cd player	4.28	70.46
11	tp laundry soap	1.20	5.17	facial cleanser	1.11	4.15	hand&body thera	0.76	5.55
12	film cameras it	1.64	5.20	planners/calend	0.94	5.02	antacid h2 bloc	0.69	3.85
13	tools/accessori	4.46	11.17	binders items 2	3.59	10.16	drawing supplie	1.96	7.71
14	american greeti	4.42	5.34	paperback items	2.69	11.04	fragrances op	2.66	12.27
15	american greeti	5.56	6.72	christmas cards	0.45	2.12	basket candy it	0.44	1.45
16	tp seasonal boo	10.78	15.49	american greeti	0.98	1.18	valentine box c	0.71	4.08
17	vitamins e item	1.76	6.79	group stationer	1.01	11.55	tp seasonal boo	0.99	1.42
18	halloween bag c	2.11	6.06	basket candy it	1.23	4.07	cold cold items	1.17	4.24
19	hair clr perman	12.00	16.76	american greeti	1.11	1.34	revlon cls face	0.83	3.07
20	revlon cls face	7.05	26.06	hair clr perman	4.14	5.77	headache ibupro	2.37	12.65

$\mathcal{C}_\ell$	top product	$	lift	sec. product	$	lift	third product	$	lift
1	action items 30	0.26	15.13	tp video comedy	0.19	15.13	family items 30	0.14	11.41
2	smoking cessati	10.15	34.73	blood pressure	1.69	34.73	snacks/pnts nut	0.44	34.73
3	underpads hea	1.31	16.52	miscellaneous k	0.53	15.59	tp irons items	0.47	14.28
4	acrylics/gels/w	0.19	11.22	tp exercise ite	0.15	11.20	dental applianc	0.81	9.50
5	appliances item	3.65	11.99	housewares peg	0.13	9.92	tp tarps items	0.22	9.58
6	multiples packs	0.17	13.87	christmas light	8.11	12.22	tv's items 6	0.44	8.32
7	sleep aids item	0.31	14.61	kava kava items	0.51	14.21	tp beer super p	0.14	12.44
8	batt rechargeab	0.34	21.82	tp razors items	0.28	21.82	tp metal cookwa	0.39	12.77
9	tp furniture it	0.45	22.42	tp art&craft al	0.19	13.77	tp family plan,	0.15	13.76
10	pers cd player	4.28	70.46	tp plumbing ite	1.71	56.24	umbrellas adult	0.89	48.92
11	cat litter scoo	0.10	8.70	child acetamino	0.12	7.25	pro treatment i	0.07	6.78
12	heaters items 8	0.16	12.91	laverdiere ca	0.14	10.49	ginseng items 4	0.20	6.10
13	mop/broom lint	0.17	13.73	halloween cards	0.30	12.39	tools/accessori	4.46	11.17
14	dental repair k	0.80	38.17	tp lawn seed it	0.44	35.88	tp telephones/a	2.20	31.73
15	gift boxes item	0.10	8.18	hearing aid bat	0.08	7.25	american greeti	5.56	6.72
16	economy diapers	0.21	17.50	tp seasonal boo	10.78	15.49	girls socks ite	0.16	12.20
17	tp wine 1.5l va	0.17	15.91	group stationer	1.01	11.55	stereos items 2	0.13	10.61
18	tp med oint liq	0.10	8.22	tp dinnerware i	0.32	7.70	tp bath towels	0.12	7.28
19	hair clr perman	12.00	16.76	covergirl imple	0.14	11.83	tp power tools	0.25	10.89
20	revlon cls face	7.05	26.06	telephones cord	0.56	25.92	ardell lashes i	0.59	21.87

Table 3.1: List of *descriptive* (top) and *discriminative* products (bottom) dominant in each of the 20 value balanced clusters obtained from the `RETAIL` data (see also figure 3.5(b)). For each item the average amount of $ spent in this cluster and the corresponding lift is given.

and cluster $\mathcal{C}_{10}$ are the big spenders, as cluster $\mathcal{C}_{10}$ is much smaller with equal revenue contribution (figure 3.5(b)). Our hunch is reinforced by looking at figure 3.5(b).

3.5.2 Web-document Clusters

In this subsection, we present results on documents from the Yahoo! news section. Each of the 2340 documents is characterized by a bag-of-words. The data is publicly available from `ftp://ftp.cs.umn.edu/dept/users/boley/` (K1 series) and was used in [BGG+99, SGM00]. The 20 original Yahoo! news categories are `Business` (B), `Entertainment` (`no sub-category` (E), `art` (a), `cable` (c), `culture` (cu), `film` (f), `industry` (i), `media` (m), `multimedia` (mm), `music` (mu), `online` (o), `people` (p), `review` (r), `stage` (s), `television` (t), `variety` (v)), `Health` (H), `Politics` (P), `Sports` (S), `Technology` (T) and correspond to the category labels 1,...,20, respectively. The raw 21839 $\times$ 2340 word-by-document matrix consists of the non-normalized occurrence frequencies of stemmed words, using Porter's suffix stripping algorithm [Fra92]. Pruning all words that occur less than 0.01 or more than 0.10 times on average because they are insignificant (e.g., `haruspex`) or too generic (e.g., `new`), respectively, results in $d = 2903$. We call this data-set `YAHOO` (see also appendix A.5).

Let us point out some worthwhile differences between clustering market-baskets and documents. Firstly, discrimination of vector length is no longer desired since customer life-time value matters but document length does not. Consequently, we use cosine similarity $s^{(C)}$ instead of extended Jaccard similarity $s^{(J)}$. Also, in document clustering we are less concerned about balancing,

since there are usually no direct monetary costs of the actions derived from the clustering involved. As a consequence of this, we over-cluster first with sample-balanced OPOSSUM and then allow user guided merging of clusters through CLUSION. The `YAHOO` data-set is notorious for having some diffuse groups with overlaps among categories, a few categories with multi-modal distributions, etc. These aspects can be easily explored by looking at the class labels within each cluster, merging some clusters and then again visualizing the results.

Figure 3.6 shows clusterings with three settings of k. For $k = 10$ (figure 3.6(a)) most clusters are not dense enough, despite the fact that the first two clusters already seem like they should not have been split. After increasing to $k = 40$ (figure 3.6(b)), CLUSION indicates that the clustering now has sufficiently compact clusters. Now, we successively merge pairs of highly related clusters until we obtain our final clustering with $k = 20$ (figure 3.6(c)). The merging process is guided by inter-cluster similarity (e.g., bright off-diagonal regions) augmented by cluster-descriptions (e.g., related frequent words). In fact, in our graphical user interface of CLUSION merging is as easy as clicking on a selected off-diagonal region.

Table 3.2(top) shows cluster evaluations, their descriptive and discriminative word stems. Each cluster ($\mathcal{C}_\ell$) is evaluated using the dominant category ($\mathcal{K}_{\hat{h}}$), purity ($\phi^{(\mathrm{A})}$), and entropy ($\phi^{(\mathrm{B})}$). Let $n_\ell^{(h)}$ denote the number of objects in cluster $\mathcal{C}_\ell$ that are classified to be in category h as given by the original Yahoo! categorization. Cluster $\mathcal{C}_\ell$'s *purity* can be defined as

$$\phi^{(\mathrm{A})}(\mathcal{C}_\ell) = \frac{1}{n_\ell} \max_h (n_\ell^{(h)}). \tag{3.9}$$

Purity can be interpreted as the classification rate under the assumption that

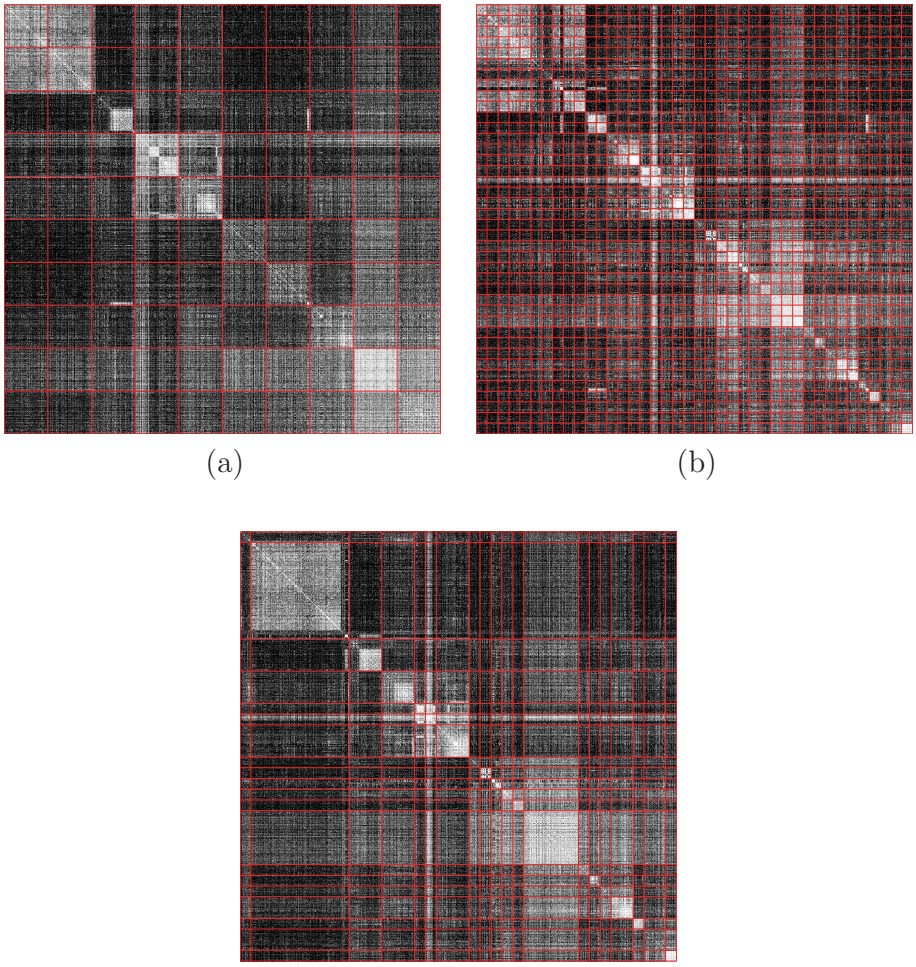


Figure 3.6: Comparison of various number of clusters k for YAHOO news data: (a) under-clustering at $k = 10$, (b) over-clustering at $k = 40$, (c) good clustering through interactive split and merge using CLUSION at $k = 20$. See color pictures in soft-copy for cluster boundaries.

$\mathcal{C}_\ell$	$\mathcal{K}_{\hat{h}}$	$\phi^{(\mathrm{A})}$	$\phi^{(\mathrm{B})}$	top 3 descriptive terms	top 3 discriminative terms
1	P	21.05%	0.73	israel, teeth, dental	mckinnei, prostat, weizman
2	H	91.48%	0.15	breast, smok, surgeri	symptom, protein, vitamin
3	S	68.39%	0.40	smith, player, coach	hingi, touchdown, rodman
4	P	52.84%	0.60	republ, committe, reform	icke, veto, teamster
5	T	63.79%	0.39	java, sun, card	nader, wireless, lucent
6	o	57.63%	0.40	apple, intel, electron	pentium, ibm, compaq
7	B	60.23%	0.48	cent, quarter, rose	dow, ahmanson, greenspan
8	f	37.93%	0.66	hbo, ali, alan	phillip, lange, wendi
9	cu	50.85%	0.48	bestsell, weekli, hardcov	hardcov, chicken, bestsell
10	p	36.21%	0.56	albert, nomin, winner	forcibl, meredith, sportscast
11	f	67.80%	0.33	miramax, chri, novel	cusack, cameron, man
12	f	77.59%	0.31	cast, shoot, indie	juliett, showtim, cast
13	r	47.28%	0.56	showbiz, sound, band	dialogu, prodigi, submiss
14	mu	44.07%	0.56	concert, artist, miami	bing, calla, goethe
15	p	50.00%	0.50	notabl, venic, classic	stamp, skelton, espn
16	mu	18.97%	0.71	fashion, sold, bbc	poetri, versac, worn
17	p	55.08%	0.54	funer, crash, royal	spencer, funer, manslaught
18	t	82.76%	0.24	househ, sitcom, timeslot	timeslot, slot, household
19	f	38.79%	0.58	king, japanes, movi	denot, winfrei, atop
20	f	69.49%	0.36	weekend, ticket, gross	weekend, gross, mimic

	B	E	a	c	cu	f	i	m	mm	mu	o	p	r	s	t	v	H	P	S	T
7	**106**	1	-	4	2	-	30	6	-	4	2	1	-	-	5	2	-	2	-	11
9	-	-	-	3	**30**	17	-	-	1	1	2	2	1	-	1	1	-	-	-	-
8	-	-	1	7	-	**22**	2	-	-	3	1	5	8	1	5	2	-	-	1	-
11	-	-	-	1	-	**40**	1	-	-	-	-	1	2	-	1	13	-	-	-	-
12	-	-	-	2	-	**45**	-	-	-	-	-	2	1	2	4	2	-	-	-	-
19	-	1	-	3	1	**45**	1	-	-	8	-	15	2	-	25	14	-	-	1	-
20	-	1	1	-	-	**41**	-	-	-	4	-	-	-	5	6	1	-	-	-	-
14	-	2	8	-	4	2	-	-	-	**26**	1	12	-	2	1	-	-	1	-	-
16	-	1	4	1	9	9	2	2	1	**11**	-	11	-	-	6	-	-	1	-	-
6	8	-	-	-	-	-	1	-	3	-	**34**	-	-	-	-	1	-	-	-	12
10	-	-	-	3	1	4	-	-	2	2	1	**21**	2	-	20	2	-	-	-	-
15	-	-	2	1	5	13	-	-	-	4	2	**29**	-	-	2	-	-	-	-	-
17	-	1	-	2	6	5	1	6	-	12	1	**65**	3	-	12	4	-	-	-	-
13	-	-	1	1	9	22	6	1	3	33	9	58	**139**	7	2	3	-	-	-	-
18	-	-	1	2	-	1	-	-	-	-	-	2	-	-	**48**	4	-	-	-	-
2	2	-	-	2	1	1	1	-	1	1	3	5	-	-	6	-	**483**	5	17	-
1	3	2	2	1	-	4	-	1	-	4	-	10	-	-	5	-	11	**12**	2	-
4	14	-	4	7	5	2	15	5	-	6	3	6	-	1	12	2	-	**93**	1	-
3	1	-	-	1	1	5	10	-	3	5	-	3	-	-	23	3	-	-	**119**	-
5	8	-	-	3	-	-	-	-	-	1	6	-	-	-	3	-	-	-	-	**37**

Table 3.2: Cluster evaluations, their descriptive and discriminative terms (top) as well as the confusion matrix (bottom) for the YAHOO news example (see also figure 3.6(c)). For each cluster number $\mathcal{C}_\ell$ the dominant category $\mathcal{K}_{\hat{h}}$, purity $\phi^{(\mathrm{A})}$, and entropy $\phi^{(\mathrm{B})}$ are shown.

all samples of a cluster are predicted to be members of the actual dominant class for that cluster. Alternatively, we also use [0,1]-normalized *entropy*, which is defined for a problem with g categories as

$$\phi^{(\mathrm{B})}(\mathcal{C}_\ell) = -\sum_{h=1}^{g} \frac{n_\ell^{(h)}}{n_\ell} \log_g \left(\frac{n_\ell^{(h)}}{n_\ell} \right). \tag{3.10}$$

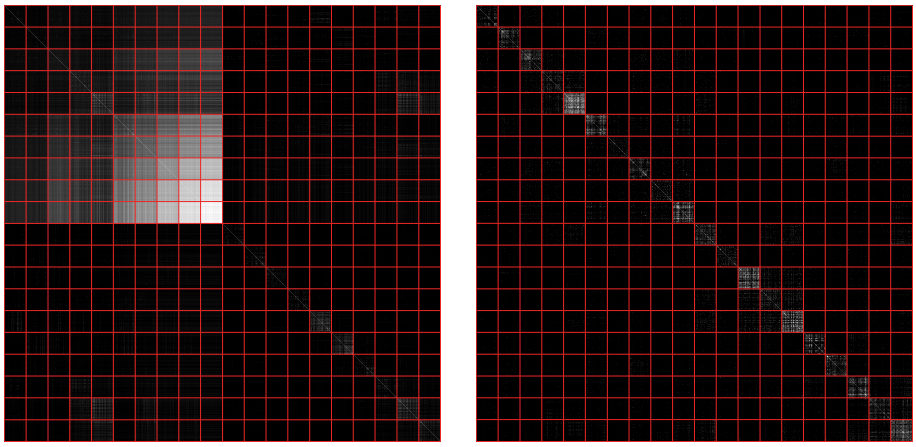
Entropy is a more comprehensive measure than purity since rather than just considering the number of objects 'in' and 'not in' the most frequent category, it considers the entire distribution. Table 3.2(bottom) gives the complete confusion matrix, which indicates how clusters and categories are related. Note that neither category nor prior distribution information is used during the unsupervised clustering process. In fact, the clustering is very good. It is much better than the original categorization in terms of edge cut and similarity lift, and it provides a much better grouping when only word frequencies are looked at. The evaluation metrics serve the purpose of validating our results capture relevant categorizations. However, their importance for our purpose is limited since we are solving a clustering problem and not a classification problem. The largest *and* best cluster is cluster $\mathcal{C}_2$ with 483 out of 528 documents being from the health cluster. Health related documents show a very distinct set of words and can, hence, be nicely separated. Small and not well distinguished categories have been put together with other documents (For example, the arts category has mostly been absorbed by the music category to form clusters 14 and 16.). This is inevitable since the 20 categories vary widely in size from 9 to 494 documents while the clusters OPOSSUM provides are much more balanced (from 58 to 528 documents per cluster).

3.5.3 Web-log Session Clusters

Web-portals and other e-commerce sites often segment their visitors to provide better personalized services. When a web-page is requested, the server log records the user's IP address, the URL retrieved, access time, etc. These logs can be analyzed to segment visitors based on their 'cow-path' or trajectory through the website, as described by the sequence of pages visited, page contents, time spent on each page, etc.

In a recent work, the use of a weighted Longest Common Subsequence (LCS) [BG01] similarity metric to describe how similar two trajectories are was suggested. This metric determines the LCS of the two trajectories, and then scales it by what fraction of the total visit time is spent in the longest common subsequence. Alternatively, one can use a vector-space model, wherein entries in the data matrix $\mathbf{X}$ indicate time spent in a particular session (column) on a particular page (row).

In this subsection, we present results of OPOSSUM and CLUSION for the data presented in [BG01]. We randomly selected 3000 sessions (out of 23310) from a community portal, `http://www.sulekha.com/`. The index / root page of the web-portal was removed since it was visited by almost everyone for a considerable amount of time and, hence, provided no discriminatory information. Figure 3.7 compares results for a vector-space-based approach using cosine similarity with LCS. The cosine measure shows some large dark diagonal regions indicating compact clusters of sessions, but it turns out that these clusters are sessions where the majority of the time was spent on a category index page (level 2 on the portal's site map). The LCS is able to capture a larger percentage of the total similarity (amount of 'grayness') in the diagonal



(a) (b)

Figure 3.7: Web-log session clustering using a vector space model and cosine similarity (a) versus using weighted longest common sub-sequence similarity (b). The cosine similarity is far less sparse and is dominated by major category index pages, while the LCS shows better isolation among the clusters.

regions, showing a better and more balanced grouping. Such visualization can be used to select the appropriate similarity measure for a given clustering objective, and to evaluate the overall clustering quality. For example, CLUSION shows that clustering visitors into 20 groups was successful despite the extreme sparsity ($\sim$1%) in figure 3.7(b). We also used value-balanced OPOSSUM to cluster web-log sessions which yields clusters with *comparable total web-surfer exposure time*. These clusters might be particularly useful for new formats in target advertising campaigns. It simplifies advertising campaign management by enabling the portal to offer fixed prizes for ad delivery exposure to each cluster since they represent comparable attention times.

3.6 System Issues

3.6.1 Synergy between OPOSSUM and CLUSION

The visualization and clustering techniques presented in this work need to be considered together and not in isolation. This is because CLUSION is particularly suited to viewing the output of OPOSSUM. First, the similarity matrix is already computed during the clustering step, so no extra computation is needed, except for permuting this matrix, which can be done in O(n) since the size and seriation order of each partition is known. Second, since METIS involves boundary Kernighan-Lin refinement, clusters that are similar appear closer in the seriation order. Thus it is no coincidence that clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ appear contiguous in figure 3.6(a). Finally, one can experiment with different similarity measures for OPOSSUM and quickly get visual feedback regarding their effectiveness using CLUSION (figure 3.7).

3.6.2 FASTOPOSSUM

Since OPOSSUM aims to achieve balanced clusters, random sampling is effective for obtaining adequate examples of each cluster. If the clusters are perfectly balanced, the distribution of the number of samples from a specific cluster in a subsample of size $\underline{n}$ taken from the entire population is binomial with mean $\underline{n}/k$ and variance $\underline{n}(k-1)/k^2$. For finite population, the variance will be even less. Thus, if we require at least r representatives from this cluster, then the number of samples is given by: $\underline{n}/k \geq z_\alpha\sqrt{\underline{n}(k-1)} + r$, where $z_\alpha = 1.96$ or 2.81 for 97.5% and 99.5% confidence levels respectively. This is $O(rk)$. For example, if we have 10 clusters and need to ensure at least 20

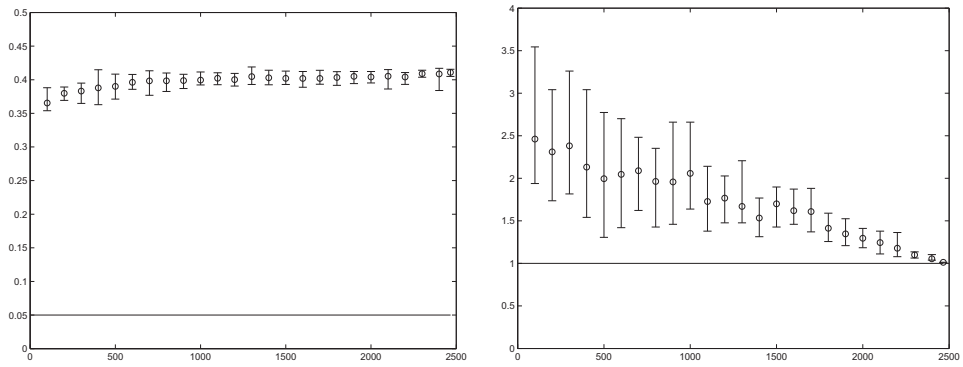
representatives from a given cluster with probability 0.995, about 400 samples are adequate. Note that this number is independent of n if n is adequately large (at least 400 in this case), so even for over one million customers, only 400 representatives are required.

This suggests a simple and effective way to scale OPOSSUM to very large number of objects n, using the following four-step process called FASTOPOSSUM:

1. Pick a boot-sample of size $\underline{n}$ so that the corresponding r value is adequate to define each cluster.
2. Apply OPOSSUM to the boot-sample to get k initial clusters.
3. Find the centroid for each of the k clusters.
4. Assign each of the remaining $n - \underline{n}$ points to the cluster with the nearest centroid.

Using $\underline{n} = \sqrt{n}$ reduces the complexity of FASTOPOSSUM to $O(kn)$. Note that the above algorithm may not result in balanced clusters. We can enforce balancing by allocating the remaining points to the k clusters in groups, each time solving a stable marriage problem [GI89], but this will increase the computation time.

Figure 3.8 illustrates the behavior of FASTOPOSSUM for the drugstore customer data-set from subsection 3.5.1. The remaining edge weight fraction indicates how much of the cumulative edge weight remains after the edge separator has been removed: $\frac{\sum_{\ell=1}^{k} \sum_{\lambda_a=\ell} \sum_{\lambda_b=\ell, b>a} s(\mathbf{x}_a, \mathbf{x}_b)}{\sum_{a=1}^{n} \sum_{b=a+1}^{n} s(\mathbf{x}_a, \mathbf{x}_b)}$ The better the partitioning, the smaller the edge-separator, and thus the larger the remaining edge weight fraction. Surprisingly the speedup does not result in a significant decreased



(a) (b)

Figure 3.8: Effect of sub-sampling on OPOSSUM. Cluster quality as measured by remaining edge weight fraction (a) and imbalance (b) of *total* graph with 2466 vertices (customers from subsection 3.5.1) for various boot-sample sizes in FASTOPOSSUM. For each setting the results' range and mean of 10 trials are depicted. Using all 2466 customers as the boot-sample (i.e., no sub-sampling) results in balancing within the 1.05 imbalance requirement and approximately 40% of edge weight remaining (as compared to 5% baseline for random clustering). As the boot-sample becomes smaller the remaining edge weight stays approximately the same (a), however the imbalance increases (b).

quality in terms of remaining edge weight (figure 3.8(a)). However, the balancing property is progressively relaxed as the boot-sample becomes smaller in comparison to the full data-set (figure 3.8(b)). Using $\underline{n} = 100$ initial points reduces the original computation time to less than 1% at comparable remaining edge weight but at an imbalance of 3.5 in the worst of 10 random trials. These results indicate that scaling to large n is easily possible, if one is willing to relax the balancedness constraints.

3.6.3 Parallel Implementation

Another notion of scalability is w.r.t. the number of processors (speedup, iso-efficiency, etc.). Our analysis [SG00b] shows almost linear speedup for our method, as the similarity computation as well as graph partitioning can both be fairly trivially parallelized with little overhead. Parallel implementation of the all-pair similarity computation on SIMD or distributed memory processors is trivial. It can be done in a systolic or block systolic manner with essentially no overhead. Frameworks such as MPI also provide native primitives for such computations. Parallelization of METIS is also very efficient, and [SKK99], reports partitioning of graphs with over 7 million vertices in 7 seconds into 128 clusters on a 128 processor Cray T3E. For further details, see [SG00b].

3.7 Summary

A recent poll (June 2001) by KDNuggets (`http://www.kdnuggets.com/`) indicated that clustering was by far the most popular type of analysis in the last 12 months at 22% (followed by direct marketing at 14% and cross-sell models at 12%). The clustering process is characterized by extensive explorative periods where better domain understanding is gained. Often, in this iterative process the crucially important definitions of features and/or similarity are refined. The visualization toolkit CLUSION allows even non-specialists to get an intuitive visual impression of the grouping nature of objects that may be originally defined in a high-dimensional space. Taking CLUSION from a post-processing step into the loop can significantly accelerate the process of discovering domain knowledge, as it provides a powerful visual aid for assess-

ing and improving clustering. For example, actionable recommendations for splitting or merging of clusters can be easily derived and readily applied via a point-and-click user interface, and different similarity metrics can be compared visually. It also guides the user towards the 'right' number of clusters. A demo of this tool can be found at `http://strehl.com/`.

This chapter originally stemmed from our encounter with several retail data-sets, where even after substantial pre-processing we were left with records with over 1000 attributes, and further attempts to reduce the number of attributes by selection/projection led to loss of vital information. Relationship-based clustering provides one way out by transforming the data to another space (in time linear in the number of dimensions) where the high dimensionality gets 'hidden', since once similarity is computed, the original dimensions are not encountered again. This suggests a connection of our approach with kernel-based methods, such as support vector machines, that are currently very popular for classification problems [Vap95, Joa98]. A kernel function of two vectors is a generalized inner product between the corresponding mappings of these vectors into a derived (and typically very high dimensional) feature space. Thus one can view it as a similarity measure between the two original vectors. It will be worthwhile to further investigate this connection for a variety of applications [JH99].

The clustering algorithm presented in this chapter is largely geared towards the needs of segmenting transactional data, with provision of getting balanced clusters and for selecting the quantity (revenue, margins) of interest to influence the grouping. Thus, rather than evaluating business objectives (such as revenue contribution) after clustering is done, they are directly integrated into the clustering algorithm. Moreover, it is a natural fit with the

visualization algorithm. Also, it can be extended to other domains, as illustrated by our results on document clustering and grouping web-logs. We also examined several ways of scaling the clustering routine to a large number of data points, and elaborated on one approach that is able to use sampling effectively because of the balanced nature of the desired clusters.

Chapter 4

Impact of Similarity Measures

I have had my results for a long time: but I do not yet
know how I am to arrive at them.
– Karl Friedrich Gauß[1]

In the last chapter, we explored the relationship-based approach to clustering in several domains. The work was initially motivated by retail data and extended naturally to other domains where high-dimensional representations are prevalent, such as text documents and web-logs. A particularly interesting application is clustering of text documents which enables unsupervised categorization and facilitates browsing and search. A critical step in adapting a relationship-based clustering to a specific domain is the choice of similarity measure. In this chapter, we investigate the impact of similarity measures on clustering quality. We will first introduce similarities and algorithms for text clustering, then develop a general comparative framework and, finally, conduct case studies on a variety of text corpora.

[1]Quoted in A. Arber, *The Mind and the Eye*, 1954

4.1 Motivation

Document clusters can provide a structure for organizing large bodies of text for efficient browsing and searching. For example, recent advances in Internet search engines (e.g., `http://vivisimo.com/`, `http://metacrawler.com/`) exploit document cluster analysis. For this purpose, a document is commonly represented as a vector consisting of the suitably normalized frequency counts of words or terms. Each document typically contains only a small percentage of all the words ever used. If we consider each document as a multi-dimensional vector and then try to cluster documents based on their word contents, the problem differs from classic clustering scenarios in several ways: Document data is high-dimensional[2], characterized by a very sparse term-document matrix with positive ordinal attribute values and a significant amount of outliers. In such situations, one is truly faced with the 'curse of dimensionality' issue [Fri94] since, even after feature reduction, one is left with hundreds of dimensions per object.

In the previous chapter, we developed the relationship-clustering framework to effectively side-step the 'curse of dimensionality'. In the relationship-based clustering process, key cluster analysis activities [JD88] can be associated with each step:

1. To obtain features $\mathbf{X} \in \mathcal{F}$ from the raw objects, a suitable *object representation* has to be found. We will not be concerned with representation in this chapter, since the significant amount of empirical studies on document clustering in the 80s and earlier emphasized various ways of

[2]The dimension of a document in vector space representation is the size of the vocabulary, often in the tens of thousands.

representing / normalizing documents [Wil88, SB88, Sal89].

2. In the second step, a *measure of proximity* $\mathbf{S} \in \mathcal{S}$ has to be defined between objects. The choice of similarity or distance can have a profound impact on clustering quality. In this chapter, we first compare similarity measures analytically and then illustrate their semantics geometrically.

3. The third activity requires a suitable choice of *clustering algorithm* to obtain cluster labels $\lambda \in \mathcal{O}$. Agglomerative clustering approaches were historically dominant as they compared favorably with flat partitional approaches on small or medium sized collections [Wil88, Ras92]. But lately, some new partitional methods have emerged (spherical k-means, graph partitioning-based, etc.) that have attractive properties in terms of both quality and scalability and can work with a wider range of similarity measures. In addition, much larger document collections are being generated.[3] This warrants an updated comparative study on text clustering, which is the motivation behind this chapter.

4. Finally, in the *assessment of output* one has to investigate the validity of the results.[4] In this chapter, we propose an experimental methodology to compare high-dimensional clusterings based on mutual information and we show how this is better than purity or entropy-based measures [BGG$^+$99, ZK01, SKK00]. Finally, we conduct a series of experiments to evaluate the performance and cluster quality of four similarity measures (Euclidean, cosine, Pearson correlation, extended Jaccard) in com-

[3]IBM Patent Server has over 20 million patents. Lexis-Nexis contains over 1 billion documents

[4]Often, *data abstraction* has to performed between clustering and final assessment [JD88].

bination with five algorithms (random, self-organizing map, hypergraph partitioning, generalized k-means, weighted graph partitioning).

Some very recent, notable comparative studies on document clustering [SKK00, ZK01] also consider some of the newer issues. Our work is distinguished from these efforts mainly by its focus on the key role of the similarity measures involved, emphasis on balancing, and the use of a normalized mutual information-based evaluation that we believe has superior properties.

The basic notation is the same as introduced in the previous chapter in section 3.2. In the next section, we introduce several similarity measures, illustrate some of their properties, and show why we are interested in some but not others. In section 4.3, the algorithms using these similarity measures are discussed. Section 4.4 introduces a variety of cluster quality evaluation methods including our proposed mutual information criterion. Finally, the experiments and results are shown in section 4.5.

4.2 Similarity Measures for Document Clustering

4.2.1 Conversion from a Distance Metric

The Minkowski distances $L_p(\mathbf{x}_a, \mathbf{x}_b) = \left(\sum_{i=1}^{d} |\mathbf{x}_{i,a} - \mathbf{x}_{i,b}|^p\right)^{1/p}$ are the standard metrics for geometrical problems. For $p = 2$ we obtain the Euclidean distance. There are several possibilities for converting such a distance metric (in $[0, \inf)$, with 0 closest) into a similarity measure (in $[0, 1]$, with 1 closest) by a monotonic decreasing function. For Euclidean space, we chose to relate dis-

tances d and similarities s using $s = e^{-d^2}$. Consequently, we define Euclidean [0,1]-normalized similarity as

$$s^{(\mathrm{E})}(\mathbf{x}_a, \mathbf{x}_b) = e^{-\|\mathbf{x}_a - \mathbf{x}_b\|_2^2} \tag{4.1}$$

which has important desirable properties (as we will see in the discussion) that the more commonly adopted $s(\mathbf{x}_a, \mathbf{x}_b) = 1/(1 + \|\mathbf{x}_a - \mathbf{x}_b\|_2)$ lacks. Other distance functions can be used as well. The Mahalanobis distance normalizes the features using the covariance matrix. Due to the high-dimensional nature of text data, covariance estimation is inaccurate and often computationally intractable, and normalization is done if need to be, at the document representation stage itself, typically by applying TF-IDF.

4.2.2 Cosine Measure

A popular measure of similarity for text (which normalizes the features by the covariance matrix) clustering is the cosine of the angle between two vectors. The cosine measure is given by

$$s^{(\mathrm{C})}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^\dagger \mathbf{x}_b}{\|\mathbf{x}_a\|_2 \cdot \|\mathbf{x}_b\|_2} \tag{4.2}$$

and captures a scale invariant understanding of similarity. An even stronger property is that the cosine similarity does not depend on the length:
$s^{(\mathrm{C})}(\alpha\mathbf{x}_a, \mathbf{x}_b) = s^{(\mathrm{C})}(\mathbf{x}_a, \mathbf{x}_b)$ for $\alpha > 0$. This allows documents with the same composition, but different totals to be treated identically which makes this the most popular measure for text documents. Also, due to this property, samples can be normalized to the unit sphere for more efficient processing [DM01].

4.2.3 Pearson Correlation

In collaborative filtering, correlation is often used to predict a feature from a highly similar mentor group of objects whose features are known. The [0,1]-normalized Pearson correlation is defined as

$$s^{(\mathrm{P})}(\mathbf{x}_a, \mathbf{x}_b) = \frac{1}{2}\left(\frac{(\mathbf{x}_a - \bar{x}_a)^\dagger(\mathbf{x}_b - \bar{x}_b)}{\|\mathbf{x}_a - \bar{x}_a\|_2 \cdot \|\mathbf{x}_b - \bar{x}_b\|_2} + 1\right), \tag{4.3}$$

where $\bar{x}$ denotes the average feature value of $\mathbf{x}$ over all dimensions. Note that this definition of Pearson correlation tends to give a full matrix. Other important correlations have been proposed, such as Spearman correlation [Spe06] which works well on rank orders.

4.2.4 Extended Jaccard Similarity

The binary Jaccard coefficient measures the degree of overlap between two sets and is computed as the ratio of the number of shared attributes (words) of $\mathbf{x}_a$ AND $\mathbf{x}_b$ to the number possessed by $\mathbf{x}_a$ OR $\mathbf{x}_b$. For example, given two sets' binary indicator vectors $\mathbf{x}_a = (0, 1, 1, 0)^\dagger$ and $\mathbf{x}_b = (1, 1, 0, 0)^\dagger$, the cardinality of their intersect is 1 and the cardinality of their union is 3, rendering their Jaccard coefficient 1/3. The binary Jaccard coefficient is often used in retail market-basket applications. In chapter 3, we extended the binary definition of Jaccard coefficient to continuous or discrete non-negative features. The extended Jaccard is computed as

$$s^{(\mathrm{J})}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^\dagger \mathbf{x}_b}{\|\mathbf{x}_a\|_2^2 + \|\mathbf{x}_b\|_2^2 - \mathbf{x}_a^\dagger \mathbf{x}_b}, \tag{4.4}$$

which is equivalent to the binary version when the feature vector entries are binary. Extended Jaccard similarity [SG00c] retains the sparsity property of

the cosine while allowing discrimination of collinear vectors as we will show in the following subsection. Another similarity measure highly related to the extended Jaccard is the Dice coefficient ($s^{(\mathrm{D})}(\mathbf{x}_a, \mathbf{x}_b) = \frac{2\mathbf{x}_a^\dagger \mathbf{x}_b}{\|\mathbf{x}_a\|_2^2 + \|\mathbf{x}_b\|_2^2}$). The Dice coefficient can be obtained from the extended Jaccard coefficient by adding $\mathbf{x}_a^\dagger \mathbf{x}_b$ to both the numerator and denominator. It is omitted here since it behaves very similar to the extended Jaccard coefficient.

4.2.5 Other (Dis-)Similarity Measures

Many other (dis-)similarity measures, such as mutual neighbor or edit distance, are possible [JMF99]. In fact, the ugly duckling theorem states [Wat69] the somewhat 'unintuitive' fact that there is no way to distinguish between two different classes of objects, when they are compared over all possible features. As a consequence, any two arbitrary objects are equally similar unless we use domain knowledge. The similarity measures discussed above are the ones deemed pertinent to text documents [Sal89, FBY92] in previous studies.

4.2.6 Discussion

Clearly, if clusters are to be meaningful, the similarity measure should be invariant to transformations natural to the problem domain. Also, normalization may strongly affect clustering in a positive or negative way. The features have to be chosen carefully to be on comparable scales and similarity has to reflect the underlying semantics for the given task.

Euclidean similarity is translation invariant but scale sensitive while cosine is translation sensitive but scale invariant. The extended Jaccard has aspects of both properties as illustrated in figure 4.1. Iso-similarity lines at

$s = 0.25$, 0.5 and 0.75 for points $\mathbf{x}_1 = (3, 1)^\dagger$ and $\mathbf{x}_2 = (1, 2)^\dagger$ are shown for Euclidean, cosine, and the extended Jaccard. For cosine similarity only the 4 (out of 12) lines that are in the positive quadrant are plotted: The two lines in the lower right part are one of two lines from $\mathbf{x}_1$ at 0.5 and 0.75. The two lines in the upper left are for $\mathbf{x}_2$ at $s = 0.5$ and 0.75. The dashed line marks the locus of equal similarity to $\mathbf{x}_1$ and $\mathbf{x}_2$ which always passes through the origin for cosine and extended Jaccard similarity.

Using Euclidean similarity $s^{(\mathrm{E})}$, iso-similarities are concentric hyperspheres around the considered point. Due to the finite range of similarity, the radius decreases hyperbolically as $s^{(\mathrm{E})}$ increases linearly. The radius does not depend on the center-point. The only location with similarity of 1 is the considered point itself and all finite locations have a similarity greater than 0. This last property tends to generate non-sparse similarity matrices. Using the cosine measure $s^{(\mathrm{C})}$ renders the iso-similarities to be hypercones all having their apex at the origin and axis aligned with the considered point. Locations with similarity 1 are on the 1-dimensional sub-space defined by this axis. The locus of points with similarity 0 is the hyperplane through the origin and perpendicular to this axis. For the extended Jaccard similarity $s^{(\mathrm{J})}$, the iso-similarities are non-concentric hyperspheres. The only location with similarity 1 is the point itself. The hypersphere radius increases with the the distance of the considered point from the origin so that longer vectors turn out to be more tolerant in terms of similarity than smaller vectors. Sphere radius also increases with similarity and as $s^{(\mathrm{J})}$ approaches 0 the radius becomes infinite rendering the sphere to the same hyperplane as obtained for cosine similarity. Thus, for $s^{(\mathrm{J})} \to 0$, extended Jaccard behaves like the cosine measure, and for $s^{(\mathrm{J})} \to 1$, it behaves like the Euclidean distance.

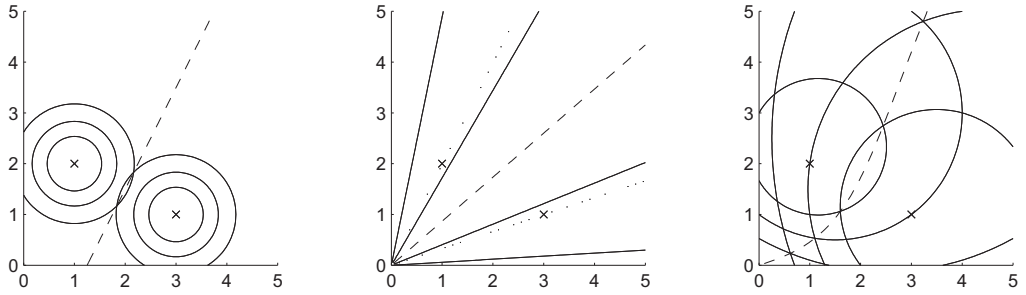


Figure 4.1: Properties of (a) Euclidean-based, (b) cosine, and (c) extended Jaccard similarity measures illustrated in 2 dimensions. Two points $(1, 2)^\dagger$ and $(3, 1)^\dagger$ are marked with $\times$s. For each point iso-similarity surfaces for $s = 0.25$, 0.5, and 0.75 are shown with solid lines. The surface that is equi-similar to the two points is marked with a dashed line.

In traditional Euclidean k-means clustering the optimal cluster representative $\mathbf{c}_\ell$ minimizes the sum of squared error criterion, i.e.,

$$\mathbf{c}_\ell = \arg\min_{\mathbf{z}\in\mathcal{F}} \sum_{\mathbf{x}_j\in\mathcal{C}_\ell} \|\mathbf{x}_j - \mathbf{z}\|_2^2. \tag{4.5}$$

In the following, we show how this convex distance-based objective can be translated and extended to similarity space. Consider the generalized objective function $f(\mathcal{C}_\ell, \mathbf{z})$ given a cluster $\mathcal{C}_\ell$ and a representative $\mathbf{z}$:

$$f(\mathcal{C}_\ell, \mathbf{z}) = \sum_{\mathbf{x}_j\in\mathcal{C}_\ell} d(\mathbf{x}_j, \mathbf{z})^2 = \sum_{\mathbf{x}_j\in\mathcal{C}_\ell} \|\mathbf{x}_j - \mathbf{z}\|_2^2. \tag{4.6}$$

We use the transformation from subsection 4.2.1 to express the objective in terms of similarity rather than distance:

$$f(\mathcal{C}_\ell, \mathbf{z}) = \sum_{\mathbf{x}_j\in\mathcal{C}_\ell} -\log(s(\mathbf{x}_j, \mathbf{z})) \tag{4.7}$$

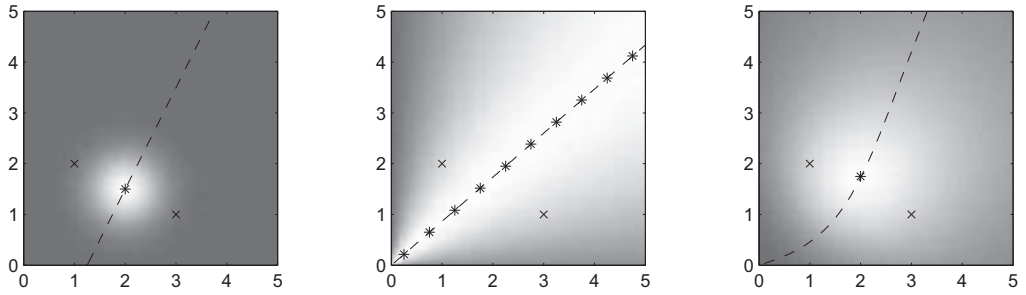


Figure 4.2: More similarity properties shown on the 2-dimensional example of figure 4.1. The goodness of a location as the common representative of the two points is indicated with brightness. The best representative is marked with a $\star$. The extended Jaccard (c) adopts the middle ground between Euclidean (a) and cosine-based similarity (b).

Finally, we simplify and transform the objective using a strictly monotonic decreasing function: Instead of minimizing $f(\mathcal{C}_\ell, \mathbf{z})$, we maximize $f'(\mathcal{C}_\ell, \mathbf{z}) = e^{-f(\mathcal{C}_\ell, \mathbf{z})}$. Thus, in similarity space, the least squared error representative $\mathbf{c}_\ell \in \mathcal{F}$ for a cluster $\mathcal{C}_\ell$ satisfies

$$\mathbf{c}_\ell = \arg\max_{\mathbf{z}\in\mathcal{F}} \prod_{\mathbf{x}_j \in \mathcal{C}_\ell} s(\mathbf{x}_j, \mathbf{z}). \tag{4.8}$$

Using the concave evaluation function f', we can obtain optimal representatives for non-Euclidean similarity spaces.

To illustrate the values of the evaluation function $f'(\{\mathbf{x}_1, \mathbf{x}_2\}, \mathbf{z})$ are used to shade the background in figure 4.2. The maximum likelihood representative of $\mathbf{x}_1$ and $\mathbf{x}_2$ is marked with a $\star$ in figure 4.2. For cosine similarity all points on the equi-similarity are optimal representatives. In a maximum likelihood interpretation, we constructed the distance similarity transformation such that $p(\mathbf{z}|\mathbf{c}_\ell) \sim s(\mathbf{z}, \mathbf{c}_\ell)$. Consequently, we can use the dual interpretations

of probabilities in similarity space and errors in distance space.

4.3 Algorithms

In this section, we briefly summarize the algorithms used in our comparison. A random algorithm is used as a baseline to compare the result quality of k-means, graph partitioning, hypergraph partitioning and self organizing maps.

4.3.1 Random Baseline (RND)

As a baseline for comparing algorithms, we use clustering labels drawn from a uniform random distribution over the integers from 1 to k. The complexity of this algorithm is $O(n)$.

4.3.2 Generalized k-means (KM)

We also employed the well-known Euclidean k-means algorithm and three variations of it using non-metric distance measures. The k-means algorithm is an iterative algorithm to minimize the least squares error criterion [DH73]. A cluster $\mathcal{C}_\ell$ is represented by its center μ_ℓ, the mean of all samples in $\mathcal{C}_\ell$. The centers are initialized with a random selection of k data objects. Each sample is then labeled with the index ℓ of the *nearest* or *most similar* center. In classical k-means, 'nearest' means the point with the smallest Euclidean distance. However, k-means can be generalized by substituting nearness with any other notion of similarity. For our comparison, we will use all four definitions of the similarity $s(\mathbf{x}_a, \mathbf{x}_b)$ between two objects $\mathbf{x}_a$ and $\mathbf{x}_b$ as introduced in the previous section. Subsequent re-computing of the mean for each cluster and

re-assigning the cluster labels is iterated until convergence to a fixed labeling after m iterations. The complexity of this algorithm is $O(n \cdot d \cdot k \cdot m)$.

4.3.3 Weighted Graph Partitioning (GP)

Clustering can be posed as a graph partitioning problem. The objects are viewed as the set of vertices $\mathcal{V}$. Two documents $\mathbf{x}_a$ and $\mathbf{x}_b$ (or vertices v_a and v_b) are connected with an undirected edge of positive weight $s(\mathbf{x}_a, \mathbf{x}_b)$, or $(a, b, s(\mathbf{x}_a, \mathbf{x}_b)) \in \mathcal{E}$. The cardinality of the set of edges $|\mathcal{E}|$ equals the number of *non-zero* similarities between all pairs of samples. A set of edges whose removal partitions a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ into k pair-wise disjoint sub-graphs $\mathcal{G}_\ell = (\mathcal{V}_\ell, \mathcal{E}_\ell)$, is called an edge separator. The objective in graph partitioning is to find such a separator with a minimum sum of edge weights. While striving for the minimum cut objective, the number of objects in each cluster has to be kept approximately equal. We produce balanced (equal sized) clusters from the similarity matrix using the multi-level graph partitioner METIS [KK98a]. The most expensive step in this $O(n^2 \cdot d)$ technique is the computation of the $n \times n$ similarity matrix. In document clustering, sparsity can be induced by looking only at the v strongest edges or at the subgraph induced by pruning all edges except the v nearest-neighbors for each vertex. Sparsity makes this approach feasible for large data-sets. Sparsity is induced by particular similarities definitions based e.g., on the cosine of document vectors.

4.3.4 Hypergraph Partitioning (HGP)

A hypergraph is a graph whose edges can connect more than two vertices (hyperedges). The clustering problem is then formulated as a finding the

minimum-cut of a hypergraph. A minimum-cut is the removal of the set of hyperedges (with minimum edge weight) that separates the hypergraph into k unconnected components. Again, an object $\mathbf{x}_j$ maps to a vertex v_j. Each word (feature) maps to a hyperedge connecting all vertices with non-zero frequency count of this word. The weight of this hyperedge is chosen to be the total number of occurrences in the data-set. Hence, the importance of a hyperedge during partitioning is proportional to the occurrence of the corresponding word. The minimum-cut of this hypergraph into k unconnected components gives the desired clustering. We employ the HMETIS package for partitioning. An advantage of this approach is that the clustering problem can be mapped to a graph problem without the explicit computation of similarity, which makes this approach computationally efficient with $O(n \cdot d \cdot k)$ assuming a (close to) linear performing hypergraph partitioner. Note that, sample-wise frequency information gets lost in this formulation since there is only a single weight associated with a hyperedge.

4.3.5 Self-Organizing Map (SOM)

The self-organizing map [Koh95, Bis95] is a popular topology preserving clustering algorithm with nice visualization properties. For simplicity, we only use a 1-dimensional line topology. Also, 2-dimensional or higher dimensional topologies can be used. To generate k clusters we use k cells in a line topology and train the network for $m = 5000$ epochs or 10 minutes (whichever comes first). All experiments are run on a dual processor 450 MHz Pentium using the SOM implementation in the Matlab neural network tool-box. The resulting network is subsequently used to generate the label vector λ from the index of

the most activated neuron for each sample. The complexity of this incremental algorithm is $O(n \cdot d \cdot k \cdot m)$ and mostly determined by the number of epochs m and samples n.

4.3.6 Other Clustering Methods

Several other clustering methods have also been considered but have not been used in our experimental comparison. Agglomerative models (single link, average link, complete link) [DH73] are computationally expensive (at least $O(n^2 \log n)$) and often result in highly skewed trees, indicating domination by one very large cluster. Mixture models [Fuk72] such as AUTOCLASS [CS96] are also popular but are limited by problems of parameter estimation for high-dimensional data. Clustering algorithms from the data mining community (e.g., CLARANS, DBSCAN, BIRCH, CLIQUE, CURE, WAVECLUSTER [RS99, HKT01]) have been omitted since they are mostly scalable versions designed for low-dimensional data. Partitioning approaches based on principal directions have not been shown here since they perform comparably to hierarchical agglomerative clustering [BGG+99]. Other graph partitioning approaches such as spectral bisectioning [HL95] are not included since they are already represented by the multi-level partitioner METIS.

4.4 Evaluation Methodology

We conducted experiments with all five algorithms, using four variants (involving different similarity measures) each for k-means and graph partitioning, yielding eleven techniques in total. This section gives an overview of ways to evaluate clustering results. A nice recent survey on clustering evaluation can

be found in [ZK01], where the emphasis is on determining the impact of a variety of cost functions, built using distance or cosine similarity measures, on the quality of two generic clustering approaches.

There are two fundamentally different ways of evaluating the quality of results delivered by a clustering algorithm. *Internal* criteria formulate quality as a function of the given data and/or similarities. For example, the mean squared error criterion is a popular evaluation criterion. Hence, the clusterer can evaluate its own performance and tune its results accordingly. When using internal criteria, clustering becomes an optimization problem. *External* criteria impose quality by additional, external information not given to the clusterer, such as class labels. While this makes the problem ill-defined, it is sometimes more appropriate since groupings are ultimately evaluated externally by humans.

4.4.1 Internal (model-based, unsupervised) Quality

Internal quality measures, such as the sum of squared errors, have traditionally been used extensively. Given an internal quality measure, clustering can be posed as an optimization problem that is typically solved via greedy search. For example, k-means has been shown to greedily optimize the sum of squared errors.

Error (mean/sum-of-squared error, scatter matrices). The most popular cost function is the scatter of the points in each cluster. Cost is measured as the mean square error of data points compared to their respective cluster centroid. The well known k-means algorithm has been shown to heuristically minimize the squared error objective. Let n_ℓ be

the number of objects in cluster $\mathcal{C}_\ell$ according to λ. Then, the cluster centroids are

$$\mathbf{c}_\ell = \frac{1}{n_\ell} \sum_{\lambda_j=\ell} \mathbf{x}_j. \tag{4.9}$$

The sum of squared errors SSE is

$$\text{SSE}(\mathbf{X}, \lambda) == \sum_{\ell=1}^{k} \sum_{\mathbf{x}\in\mathcal{C}_\ell} \|\mathbf{x} - \mathbf{c}_\ell\|_2^2. \tag{4.10}$$

Note that the SSE formulation can be extended to other similarities by using $\text{SSE}(\mathbf{X}, \lambda) = \sum_{\ell=1}^{k} \sum_{\mathbf{x}\in\mathcal{C}_\ell} - \log s(\mathbf{x}, \mathbf{c}_\ell)$. Since, we are interested in a quality measure ranging from 0 to 1, where 1 indicates a perfect clustering, we define quality as

$$\phi^{(\text{S})}(\mathbf{X}, \lambda) = e^{-\text{SSE}(\mathbf{X},\lambda)}. \tag{4.11}$$

This objective can also be viewed from a probability density estimation perspective using EM [DLR77]. Assuming the data is generated by a mixture of multi-variate Gaussians with identical, diagonal covariance matrices, the SSE objective is equivalent to the maximizing the likelihood of observing the data by adjusting the centers and minimizing weights of the Gaussian mixture.

Edge cut. When clustering is posed as a graph partitioning problem, the objective is to minimize edge cut. Formulated as a [0,1]-quality maximization problem, the objective is the ratio of remaining edge weights to total pre-cut edge weights:

$$\phi^{(\text{C})}(\mathbf{X}, \lambda) = \frac{\sum_{\ell=1}^{k} \sum_{a\in\mathcal{C}_\ell} \sum_{b\in\mathcal{C}_\ell, b>a} s(\mathbf{x}_a, \mathbf{x}_b)}{\sum_{a=1}^{n} \sum_{b=a+1}^{n} s(\mathbf{x}_a, \mathbf{x}_b)} \tag{4.12}$$

Note that this quality measure can be trivially maximized when there are no restrictions on the sizes of clusters. In other words, edge cut quality evaluation is only fair when the compared clusterings are well-balanced. Let us define the balance of a clustering λ as

$$\phi^{(\mathrm{BAL})}(\lambda) = \frac{n/k}{\max_{\ell \in \{1,\ldots,k\}} n_\ell}. \tag{4.13}$$

A balance of 1 indicates that all clusters have the same size. In certain applications, balanced clusters are desirable because each cluster represents an equally important share of the data. Balancing has application driven advantages e.g., for distribution, navigation, summarization of the clustered objects. In chapter 3, retail customer clusters are balanced so they represent an equal share of revenue. Balanced clustering for browsing text documents have also been proposed [BG02]. However, some natural classes may not be of equal size, so relaxed balancing may become necessary. A middle ground between no constraints on balancing (e.g., k-means) and tight balancing (e.g., graph partitioning) can be achieved by over-clustering using a balanced algorithm and then merging clusters subsequently [KHK99].

Category Utility [GC85, Fis87a]. The category utility function measures quality as the increase in predictability of attributes given a clustering. Category utility is defined as the *increase* in the expected number of attribute values that can be correctly guessed given a partitioning, *over* the expected number of correct guesses with no such knowledge. A weighted average over categories allows comparison of different sized partitions. For binary features (i.e., attributes) the probability of the i-th attribute

to be 1 is the mean of the i-th row of the data matrix $\mathbf{X}$:

$$\bar{x}_i = \frac{1}{n} \sum_{j=1}^{n} x_{i,j} \tag{4.14}$$

The conditional probability of the i-th attribute to be 1 given that the data point is in cluster ℓ is

$$\bar{x}_{i,\ell} = \frac{1}{n_\ell} \sum_{\lambda_j=\ell} x_{i,j}. \tag{4.15}$$

Hence, category utility can be written as

$$\phi^{(\mathrm{CU})}(\mathbf{X}, \lambda) = \frac{4}{d} \sum_{\ell=1}^{k} \frac{n_\ell}{n} \left[\left(\sum_{i=1}^{d} \left(\bar{x}_{i,\ell}^2 - \bar{x}_{i,\ell} \right) \right) - \left(\sum_{i=1}^{d} \left(\bar{x}_i^2 - \bar{x}_i \right) \right) \right]. \tag{4.16}$$

Note that our definition divides the standard category by d so that $\phi^{(\mathrm{CU})}$ never exceeds 1. Recently, it has been shown that category utility is related to squared error criterion for a particular standard encoding [Mir01]. Category utility is defined to maximize predictability of attributes for a clustering. This limits the scope of this quality measure to low-dimensional clustering problems (preferably with each dimension being a categorical variable with small cardinality). In high-dimensional problems, such as text clustering, the objective is *not* to be able to predict the appearance of any possible word in a document from a particular cluster. In fact, there might be more unique words / terms / phrases than documents in a small data-set. In preliminary experiments, category utility did not succeed in differentiating among the compared approaches (including random partitioning).

Using internal quality measures, fair comparisons can only be made amongst clusterings with the same choices of vector representation and similarity /

distance measure. E.g., using edge-cut in cosine-based similarity would not be meaningful for an evaluation of Euclidean k-means. So, in many applications a consensus on the internal quality measure for clustering is not found. However, in situations where the pages are categorized (labelled) by an external source, there is a plausible way out!

4.4.2 External (model-free, semi-supervised) Quality

External quality measures require an external grouping, for example as indicated by category labels, that is assumed to be 'correct'. However, unlike in classification such ground truth is not available to the clustering algorithm. This class of evaluation measures can be used to compare start-to-end performance of any kind of clustering regardless of the models or similarities used. However, since clustering is an unsupervised problem, the performance cannot be judged with the same certitude as for a classification problem. The external categorization might not be optimal at all. For example, the way web-pages are organized in the Yahoo! taxonomy is certainly not the best structure possible. However, achieving a grouping similar to the Yahoo! taxonomy is certainly indicative of successful clustering.

Given g categories (or classes) $\mathcal{K}_h$ ($h \in \{1, \ldots, g\}$), we denote the categorization label vector κ, where $x_a \in \mathcal{K}_h \Leftrightarrow \kappa_a = h$. Let $n^{(h)}$ be the number of objects in category $\mathcal{K}_h$ according to κ, and n_ℓ the number of objects in cluster $\mathcal{C}_\ell$ according to λ. Let $n_\ell^{(h)}$ denote the number of objects that are in cluster ℓ according to λ as well as in category h given by κ. There are several ways of comparing the class labels with cluster labels:

Purity. Purity can be interpreted as classification accuracy under the as-

sumption that all objects of a cluster are classified to be members of the dominant class for that cluster. For a single cluster $\mathcal{C}_\ell$, purity is defined as the ratio of the number of objects in the *dominant* category to the total number of objects:

$$\phi^{(\mathrm{A})}(\mathcal{C}_\ell, \kappa) = \frac{1}{n_\ell} \max_h (n_\ell^{(h)}) \tag{4.17}$$

To evaluate an entire clustering, one computes the average of the cluster-wise purities weighted by cluster size:

$$\phi^{(\mathrm{A})}(\lambda, \kappa) = \frac{1}{n} \sum_{\ell=1}^{k} \max_h (n_\ell^{(h)}) \tag{4.18}$$

Entropy [CT91]. Entropy is a more comprehensive measure than purity since rather than just considering the number of objects 'in' and 'not in' the dominant class, it takes the entire distribution into account. Since a cluster with all objects from the same category has an entropy of 0, we define entropy-based quality as 1 minus the [0,1]-normalized entropy. We define entropy-based quality for each cluster as:

$$\phi^{(\mathrm{E})}(\mathcal{C}_\ell, \kappa) = 1 - \sum_{h=1}^{g} -\frac{n_\ell^{(h)}}{n_\ell} \log_g \left(\frac{n_\ell^{(h)}}{n_\ell} \right) \tag{4.19}$$

And through weighted averaging, the total entropy quality measure falls out to be:

$$\phi^{(\mathrm{E})}(\lambda, \kappa) = 1 + \frac{1}{n} \sum_{\ell=1}^{k} \sum_{h=1}^{g} n_\ell^{(h)} \log_g \left(\frac{n_\ell^{(h)}}{n_\ell} \right) \tag{4.20}$$

Both, purity and entropy are biased to favor large number of clusters. In fact, for both these criteria, the globally optimal value is trivially reached when each cluster is a single object!

Precision, recall & F-measure [vR79]. Precision and recall are standard measures in the information retrieval community. If a cluster is viewed as the results of a query for a particular category, then precision is the fraction of correctly retrieved objects:

$$\phi^{(\mathrm{P})}(\mathcal{C}_\ell, \mathcal{K}_h) = n_\ell^{(h)} / n_\ell \tag{4.21}$$

Recall is the fraction of correctly retrieved objects out of all matching objects in the database:

$$\phi^{(\mathrm{R})}(\mathcal{C}_\ell, \mathcal{K}_h) = n_\ell^{(h)} / n^{(h)} \tag{4.22}$$

The F-measure combines precision and recall into a single number given a weighting factor. The F_1-measure combines precision and recall with equal weights. The following equation gives the F_1-measure when querying for a particular category $\mathcal{K}_h$

$$\phi^{(\mathrm{F}_1)}(\mathcal{K}_h) = \max_\ell \frac{2\ \phi^{(\mathrm{P})}(\mathcal{C}_\ell, \mathcal{K}_h)\ \phi^{(\mathrm{R})}(\mathcal{C}_\ell, \mathcal{K}_h)}{\phi^{(\mathrm{P})}(\mathcal{C}_\ell, \mathcal{K}_h) + \phi^{(\mathrm{R})}(\mathcal{C}_\ell, \mathcal{K}_h)} = \max_\ell \frac{2n_\ell^{(h)}}{n_\ell + n^{(h)}} \tag{4.23}$$

Hence, for the entire clustering the total F_1-measure is:

$$\phi^{(\mathrm{F}_1)}(\lambda, \kappa) = \frac{1}{n} \sum_{h=1}^{g} n^{(h)} \phi^{(\mathrm{F})}(\mathcal{K}_h) = \frac{1}{n} \sum_{h=1}^{g} n^{(h)} \max_\ell \frac{2n_\ell^{(h)}}{n_\ell + n^{(h)}} \tag{4.24}$$

Unlike purity and entropy, the F_1-measure is not biased towards a larger number of clusters. In fact, it favors coarser clusterings. Another issue is that random clustering tends not to be evaluated at 0.

Mutual information [CT91]. The mutual information is the most theoretically well-founded among the considered quality measures. It is unbiased

and symmetric in terms of κ and λ. We propose a [0,1]-normalized mutual information-based quality criterion $\phi^{(\mathrm{NMI})}$ which can be computed as follows:

$$\phi^{(\mathrm{NMI})}(\lambda,\kappa) = \frac{2}{n}\sum_{\ell=1}^{k}\sum_{h=1}^{g} n_\ell^{(h)} \log_{k\cdot g}\left(\frac{n_\ell^{(h)} n}{n^{(h)} n_\ell}\right) \tag{4.25}$$

A detailed derivation of this definition can be found in appendix B.1. Mutual information does not suffer from the biases like purity, entropy and the F_1-measure. Singletons are not evaluated as perfect. Random clustering has mutual information of 0 in the limit. However, the best possible labeling evaluates to less than 1, unless classes are balanced.[5] Note that our normalization penalizes over-refinements unlike the standard mutual information.[6]

External criteria enable us to compare different clustering methods fairly provided the external ground truth is of good quality. One could argue against external criteria that clustering does not have to perform as well as classification. However, in many cases clustering is an interim step to better understand and characterize a complex data-set before further analysis and modeling.

Normalized mutual information will be our preferred choice of evaluation in the next section, because it is an unbiased measure for the usefulness of the knowledge captured in the clustering in predicting category labels.

[5]There are other ways of normalizing mutual information such that the best possible labeling evaluates to 1 even when the categorization is not balanced. However, these tend to biased towards high k. Details can be found in appendices B.1 and B.2.

[6]Let $\kappa = (1,1,2,2)^\dagger$, $\lambda^{(1)} = (1,1,2,2)^\dagger$, and $\lambda^{(2)} = (1,2,3,4)^\dagger$. $\lambda^{(2)}$ is an over-refinement of correct clustering $\lambda^{(1)}$. The mutual information between κ and $\lambda^{(1)}$ is 2 and the mutual information between κ and $\lambda^{(2)}$ is 2, also. Our [0,1]-normalized mutual information measure $\phi^{(\mathrm{NMI})}$ penalizes the useless refinement: $\phi^{(\mathrm{NMI})}(\lambda^{(2)},\kappa) = 2/3$ which is less than $\phi^{(\mathrm{NMI})}(\lambda^{(1)},\kappa) = 1$.

4.5 Experiments on Text Documents

4.5.1 Data-sets and Preprocessing

We chose four text data-sets for comparison. In this subsection we will briefly describe them:

- `YAHOO`. This data was parsed from Yahoo! news web-pages [BGG+99]. The 20 original categories for the pages are:
  - `Business`
  - `Entertainment`
    - `no sub-category`
    - `art`
    - `cable`
    - `culture`
    - `film`
    - `industry`
    - `media`
    - `multimedia`
    - `music`
    - `online`
    - `people`
    - `review`
    - `stage`
    - `television`

    * variety
  - Health
  - Politics
  - Sports
  - Technology

  The data can be downloaded from ftp://ftp.cs.umn.edu/dept/users/boley/ (K1 series) (see also appendix A.5).

- N20. The data contains roughly 1000 postings each from the following 20 newsgroup topics [Lan95]:
  - alt.atheism
  - comp.graphics
  - comp.os.ms-windows.misc
  - comp.sys.ibm.pc.hardware
  - comp.sys.mac.hardware
  - comp.windows.x
  - misc.forsale
  - rec.autos
  - rec.motorcycles
  - rec.sport.baseball
  - rec.sport.hockey
  - sci.crypt

  - sci.med
  - sci.electronics
  - sci.space
  - soc.religion.christian
  - talk.politics.guns
  - talk.politics.mideast
  - talk.politics.misc
  - talk.religion.misc

  The data can be found e.g., at http://www.at.mit.edu/∼jrennie/ /20Newsgroups/ (see also appendix A.6).

- WEBKB. From the CMU Web KB Project [CDF+98], web-pages from the following ten industry sectors according to Yahoo! were selected:
  - airline
  - computer hardware
  - electronic instruments and controls
  - forestry and wood products
  - gold and silver
  - mobile homes and rvs
  - oil well services and equipment
  - railroad
  - software and programming

– trucking

Each industry contributes about 10% of the pages.

- REUT. The Reuters-21578, Distribution 1.0 is available from Lewis at http://www.research.att.com/~lewis/. We use the primary topic keyword as the category. There are 82 unique primary topics in the data. The categories are highly imbalanced.

The data-sets encompass a large variety of text styles. E.g., in WEBKB documents vary significantly in length, some are in the wrong category, some are dead links or have little content (e.g., are mostly images). Also, the hub pages that Yahoo! refers to are usually top-level branch pages. These tend to have more similar bag-of-words content across different classes (e.g., contact information, search windows, welcome messages) than news content oriented pages. In contrast, the content of REUT are well written news agency messages. However, they often belong to more than one category.

Words were stemmed using Porter's suffix stripping algorithm [FBY92] in YAHOO and REUT. For all data-sets, words occurring on average between 0.01 and 0.1 times per document were counted to yield the term-document matrix. This excludes stop words such as a, and very generic words such as new, as well as too rare words such as haruspex.

4.5.2 Results

In this section, we present and compare the results of the eleven approaches on the four document data-sets. Clustering quality is understood in terms of mutual information and balance. For each data-set we set the number of

clusters k to be twice the number of categories g, except for the REUT data-set where we used $k = 40$ since there are many small categories. Using a greater number of clusters than classes allows multi-modal distributions for each class. For example, in an XOR like problem, there are two classes, but four clusters.

Let us first look at a representative result to illustrate the behavior of some algorithms and our evaluation methodology. In figure 4.3, confusion matrices illustrating quality differences of RND, KM E, KM C, and GP C approaches on a sample of 800 documents from N20 are shown. The horizontal and vertical axis correspond to the categories and clusters, respectively. Clusters are sorted in increasing order of dominant category. Entries indicate the number $n_\ell^{(h)}$ of documents in cluster ℓ and category h by darkness. Expectedly, random partitioning RND results in indiscriminating clusters with a mutual information score $\phi^{(\mathrm{NMI})} = 0.16$. The purity score $\phi^{(\mathrm{A})} = 0.16$ indicates that on average the dominant category contributes 16% of the objects in a cluster. However, since labels are drawn from a uniform distribution, cluster sizes are somewhat balanced with $\phi^{(\mathrm{BAL})} = 0.63$. KM E delivers one large cluster (cluster 15) and many small clusters with $\phi^{(\mathrm{BAL})} = 0.03$. This strongly imbalanced clustering is characteristic for KM E on high-dimensional sparse data and is problematic because it usually defeats certain application specific purposes such as browsing. It also results in sub-random quality $\phi^{(\mathrm{NMI})} = 0.11$ ($\phi^{(\mathrm{A})} = 0.17$). KM C results are good. A 'diagonal' can be clearly seen in the confusion matrix. This indicates that the clusters align with the ground truth categorization which is reflected by an overall mutual information $\phi^{(\mathrm{NMI})} = 0.35$ ($\phi^{(\mathrm{A})} = 0.38$). Balancing is good as well with $\phi^{(\mathrm{BAL})} = 0.45$. GP C exceeds KM C in both aspects with $\phi^{(\mathrm{NMI})} = 0.47$ ($\phi^{(\mathrm{A})} = 0.48$) as well as balance $\phi^{(\mathrm{BAL})} = 0.95$. The 'diagonal' is stronger and

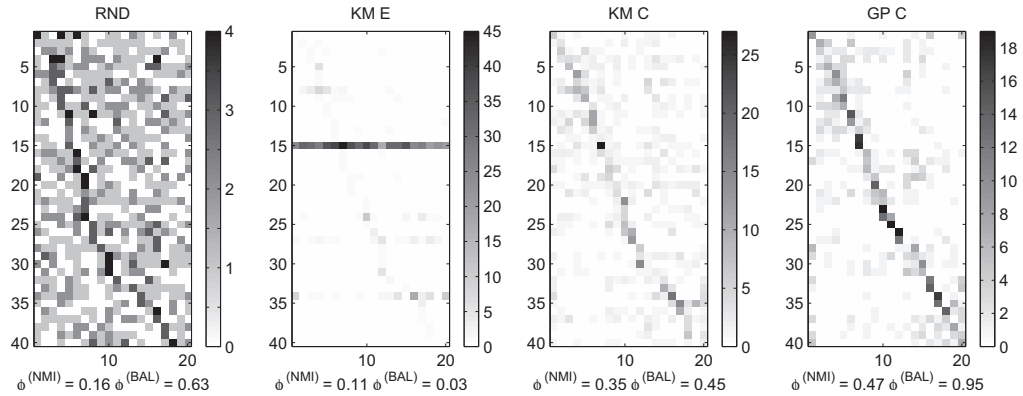


Figure 4.3: Confusion matrices illustrating quality differences of RND, KM E, KM C, and GP C approaches on a sample of 800 documents from `N20`. Matrix entries indicate the number $n_\ell^{(h)}$ of documents in cluster ℓ (row) and category h (column) by darkness. Clusters are sorted in ascending order of their dominant category. KM E delivers one large cluster and shows sub-random quality $\phi^{(\mathrm{NMI})}$. KM C results are good, but are exceeded by GP C in terms of mutual information $\phi^{(\mathrm{NMI})}$ as well as balance $\phi^{(\mathrm{BAL})}$.

clusters are very balanced.

The rest of the results are given in summarized form instead of the more detailed treatment in the example above, since the comparative trends are very clear even at this macro level. Some examples of detailed confusion matrices and pairwise t-tests can be found in our earlier work [SGM00].

For a systematic comparison, ten experiments were performed for each of the random samples of sizes 50, 100, 200, 400, and 800. Figure 4.4 shows performance curves in terms of (relative) mutual information comparing 10 algorithms on 4 data sets. Each curve shows the *difference* $\Delta\phi^{(\mathrm{NMI})}$ in mutual information-based quality $\phi^{(\mathrm{NMI})}$ compared to random partitioning for 5 sample sizes (at 50, 100, 200, 400, and 800). Error bars indicate ± 1 standard

deviations over 10 experiments. Figure 4.5 shows quality in terms of balance for 4 data sets in combination with 10 algorithms. Each curve shows the cluster balance $\phi^{(\mathrm{BAL})}$ for 5 sample sizes (again at 50, 100, 200, 400, and 800). Error bars indicate ± 1 standard deviations over 10 experiments. Figure 4.6 summarizes the results on all four data-sets at the highest sample size level ($n = 800$). We also conducted pairwise t-tests at $n = 800$ to assure differences in average performance are significant. For illustration and brevity, we chose to show mean performance with standard variation bars rather than the t-test results (see our previous work [SGM00]).

First, we look at quality in terms of mutual information (figures 4.4, 4.6(a)). With increasing sample size n, the quality of clusterings tends to improve. Non-metric (cosine, correlation, Jaccard) graph partitioning approaches work best on text data ($n = 800 : \Delta\phi^{(\mathrm{NMI})} \approx 0.3$) followed by non-metric k-means approaches. Clearly, a non-metric e.g., dot-product-based similarity measure is necessary for good quality. Due to the conservative normalization, depending on the given data-set the maximum obtainable mutual information (for a perfect *classifier*!) tends to be around 0.8 to 0.9. A mutual information-based quality around 0.4 and 0.5 (which is approximately 0.3 to 0.4 better than random at $n = 800$) is an excellent result.[7] Hypergraph partitioning constitutes the third tier. Euclidean techniques including SOM perform rather poorly. Surprisingly, the SOM still delivers significantly better than random results despite the limited expressiveness of the implicitly used Euclidean distances. The success of SOM is explained with the fact that the Euclidean distance becomes locally meaningful once the cell-centroids are locked onto a

[7]For verification purposes we also computed entropy values for our experiments and compared with e.g., [ZK01] to ensure validity.

good cluster.

All approaches behaved consistently over the four data-sets with only slightly different scale caused by the different data-sets' complexities. The performance was best on YAHOO and WEBKB followed by N20 and REUT. Interestingly, the gap between GP and KM techniques is wider on YAHOO than on WEBKB. The low performance on REUT is probably due to the high number of classes (82) and their widely varying sizes.

In order to assess which approaches are more suitable for a particular amount of objects n, we also looked for intersects in the performance curves of the top algorithms (non-metric GP and KM, HGP).[8] In our experiments, the curves do *not* intersect indicating that ranking of the top performers does not change in the range of data-set sizes considered.

In terms of balance (figures 4.5, 4.6(b)) the advantages of graph partitioning are clear. Graph partitioning explicitly tries to achieve balanced clusters ($n = 800 : \phi^{(\mathrm{BAL})} \approx 0.9$). The second tier is hypergraph partitioning which is also a balanced technique ($n = 800 : \phi^{(\mathrm{BAL})} \approx 0.7$) followed by non-metric k-means approaches ($n = 800 : \phi^{(\mathrm{BAL})} \approx 0.5$). Poor balancing is shown by SOM and Euclidean k-means ($n = 800 : \phi^{(\mathrm{BAL})} \approx 0.1$). Interestingly, balancedness does not change significantly for the k-means-based approaches as the number of samples n increases. Graph partitioning-based approaches quickly approach perfect balancing as would be expected since they are explicitly designed to do so.

Non-metric graph partitioning is significantly better in terms of mutual information as well as in balance. There is no significant difference in perfor-

[8] Intersections of performance curves in classification (learning curves) have been studied recently e.g., in [PPS01].

mance amongst the non-metric similarity measures using cosine, correlation, and extended Jaccard. Euclidean distance-based approaches do not perform better than random clustering.

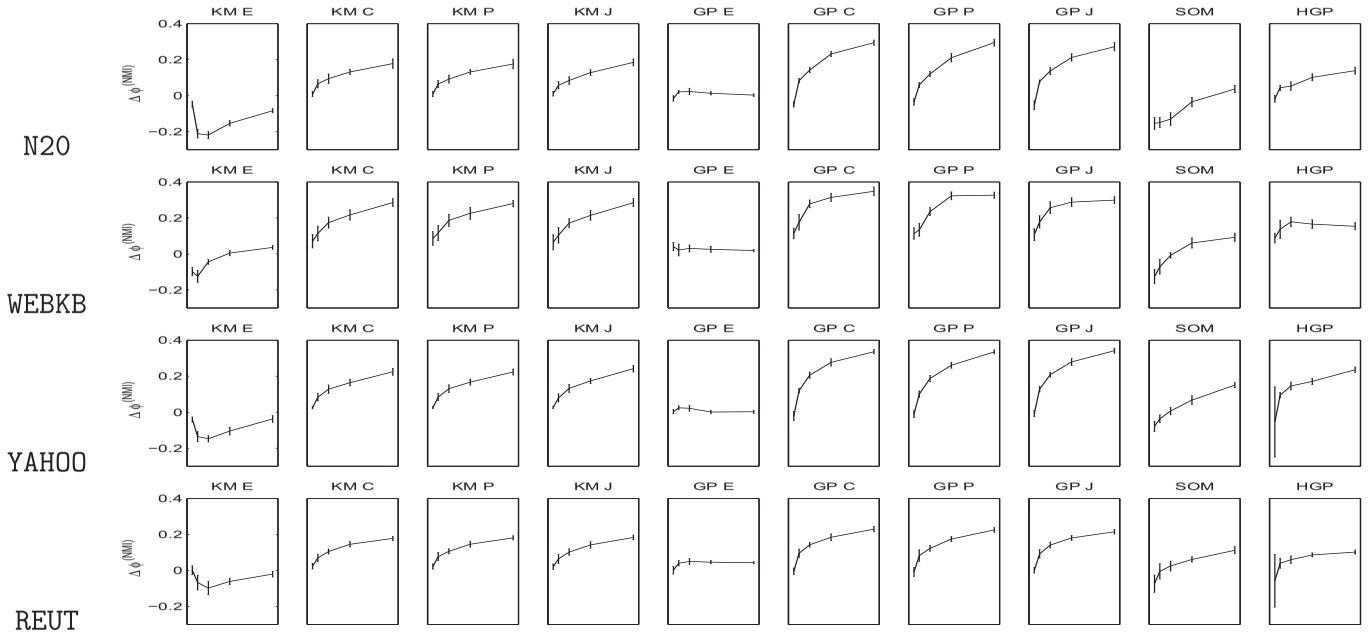


Figure 4.4: Mutual information performance curves comparing 10 algorithms on 4 data-sets. Each curve shows the difference in mutual information-based quality $\phi^{(\mathrm{NMI})}$ compared to random for 5 sample sizes (at 50, 100, 200, 400, and 800). Error bars indicate ± 1 standard deviations over 10 experiments.

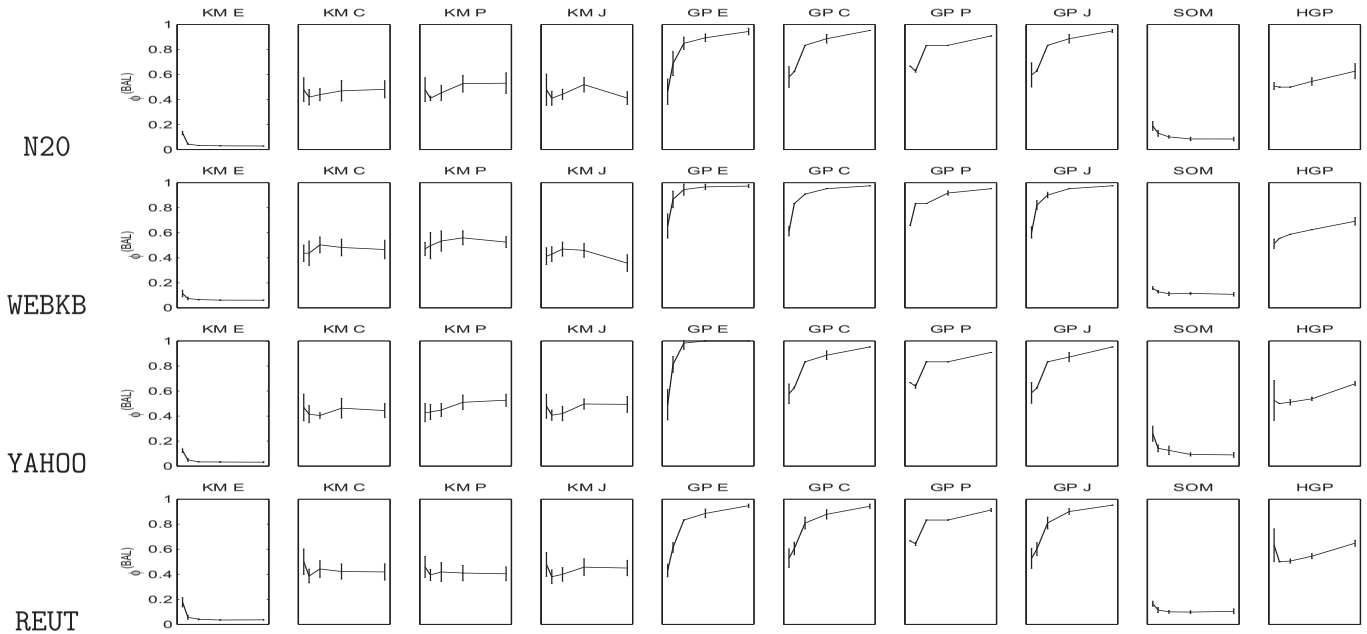
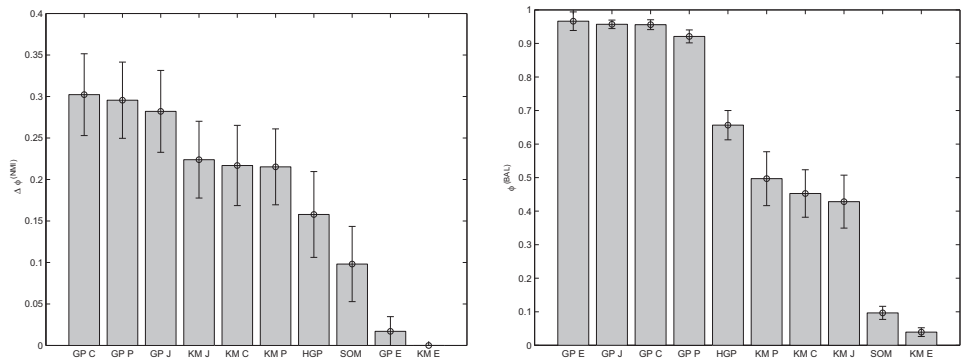


Figure 4.5: Amount of balancing achieved for 4 data-sets in combination with 10 algorithms. Each curve shows the cluster balance $\phi^{(\mathrm{BAL})}$ for 5 sample sizes (at 50, 100, 200, 400, and 800). Error bars indicate ± 1 standard deviations over 10 experiments.



(a) (b)

Figure 4.6: Comparison of cluster quality in terms of (a) mutual information and (b) balance on average over 4 data-sets with 10 trials each at 800 samples. Error-bars indicate ± 1 standard deviation. Graph partitioning is significantly better in terms of mutual information as well as in balance. Euclidean distance-based approaches do not perform better than random clustering.

4.6 Summary

The key contributions of this chapter lie in highlighting the advantages of working in a similarity space when clustering very high-dimensional sparse data such as text documents, and in providing a framework for comparing several clustering approaches *across* a variety of similarity spaces using mutual information. Another useful contribution is the conceptual assessment of a variety of similarity measures and evaluation criteria.

The comparative results indicate that for word frequency-based clustering of web documents, graph partitioning is better suited than generalized k-means, hypergraph partitioning, and SOM. The search procedure implicit in graph partitioning is far less local than the hill-climbing approach of k-means.

Moreover, it also provides a way to obtain balanced clusters and exhibit a lower variance in results.

Metric distances (such as Euclidean distance) are not appropriate for high-dimensional, sparse domains. Cosine, correlation and extended Jaccard measures are successful and perform equivalently in capturing the similarities implicitly indicated by manual categorizations.

Chapter 5

Cluster Ensembles

> The whole is more than the sum of its parts.
>
> – Aristotle[1]

It is widely recognized that combining multiple classification or regression models typically provides superior results compared to using a single, well-tuned model. However, there are no well known approaches to combining multiple non-hierarchical clusterings. The idea of combining object partitionings *without* accessing the original objects' features leads us to a general knowledge reuse framework that we call *cluster ensembles*. Our contribution in this chapter is to formally define the cluster ensemble problem as an optimization problem in terms of mutual information and to propose three effective and efficient combiners (consensus functions) for solving it. The combiners are designed with the relationship-based approach developed in this dissertation, because similarity can be used naturally in the label space to infer the relationships between clusters and/or objects. The first combiner induces a pairwise

[1] In *Metaphysica*

similarity measure between objects from the partitionings and then reclusters the objects. The second combiner creates multi-fold relationships (hyperedges) amongst the objects and uses them to repartition based on hypergraph partitioning. The third one uses the similarity of labels to group clusters into meta-clusters. Collapsed meta-clusters then compete for each object to determine the combined clustering. We also compare the approaches in a controlled experiment and propose a supra-consensus function that combines all three. We present three situations where our combiners can be used as wrappers to integrate sets of groupings: robust centralized clustering, object-distributed clustering, and feature-distributed clustering. Results on synthetic as well as real web data-sets are given to show that cluster ensembles can: (i) improve quality and robustness, (ii) enable distributed clustering, and (iii) speed up processing significantly with little loss in quality.

5.1 Motivation

The notion of integrating multiple data sources and/or learned models is found in several disciplines, for example, the combining of estimators in econometrics [Gra89], evidences in rule-based systems [Bar81] and multi-sensor data fusion [Das94]. A simple but effective type of such multi-learner systems are *ensembles*, wherein each component learner (typically a regressor or classifier) tries to solve the same task. Each learner may receive somewhat different subsets of the data for 'training' or parameter estimation (as in bagging [Bre94] and boosting [DCJ+94]), and may use different feature extractors on the same raw data. The system output is determined by *combining* the outputs of the individual learners via a variety of methods including voting,

(weighted) averaging, order statistics, product rule, entropy, and stacking. In the past few years, a host of experimental results have shown that such ensembles provide statistically significant improvements in performance along with tighter confidence intervals, especially when the component models are 'unstable' [Wol92, GBC92, Sha96, CS95]. Moreover, theoretical analysis has been developed for both regression [Per93, Has93] and classification [TG96] to estimate the gains achievable. Combining is an effective way of reducing model variance, and in certain situations it also reduces bias [Per93, TG99]. It works best when each learner is well trained, but different learners generalize in different ways, i.e., there is diversity in the ensemble [KV95, Die01].

In unsupervised learning, the clustering problem is concerned with partitioning a set of objects $\mathcal{X}$ into k disjoint[2] groups / clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$ such that similar objects are in the same cluster while dissimilar objects are in different clusters. Clustering is often posed as an optimization problem by defining a suitable error criterion to be minimized. Using this formulation, many standard algorithms such as k-means and agglomerative clustering have been established over the past forty years [Eve74, JD88]. Recent interest in the data mining community has lead to a new breed of approaches such as CLARANS, DBScan, BIRCH, CLIQUE, and WaveCluster [RS99], that are oriented towards clustering large data-sets kept in databases and emphasize scalability.

Unlike classification problems, there are no well known approaches to combining multiple clusterings. We call the *combination* of multiple *partitionings* of the same underlying set of objects *without accessing the original*

[2]The methods can be extended to combine soft clusterings, wherein an object can belong to multiple clusters with different degrees of 'association'.

features as the *cluster ensemble* design problem. Since the combiner has no more access to the original features and only cluster labels are available, this is a framework for knowledge reuse [BG99]. This problem is more difficult than designing *classifier ensembles* since cluster labels are symbolic and so one must also solve a correspondence problem. In addition, the number and shape of clusters provided by the individual solutions may vary based on the clustering method as well as on the particular view of the data presented to that method. Moreover, the desired number of clusters is often not known in advance. In fact, the 'right' number of clusters in a data-set depends on the *scale* at which the data is inspected, and sometimes, equally valid (but substantially different) answers can be obtained for the same data [CG96].

A *clusterer* consists of a particular clustering algorithm with a specific view of the data. A *clustering* is the output of a clusterer and consists of the group labels for some or all objects (a *labeling*). Cluster ensembles provide a tool for consolidation of results from a portfolio of individual clustering results. It is useful in a variety of contexts:

- *Quality and Robustness.* Combining several clusterings can lead to *improved quality* and robustness of results. For classification or regression problems, it has been shown the gains from using ensemble methods are directly related to the amount of diversity among the individual component models [KV95, TG99]. One desires that each individual model be good, but at the same time, these models should have different inductive biases and thus generalize in distinct ways [Die01]. No wonder ensembles are most popular for integrating relatively unstable models such as decision trees and multi-layered perceptrons. In the clustering context, diversity can be created in numerous ways, including:

- using different features to represent the objects. For example, images can be represented by their pixels, histograms, location and parameters of perceptual primitives, 3D scene coordinates, etc.
- varying the number and/or location of initial cluster centers in iterative algorithms such as k-means,
- varying the order of data presentation in on-line methods such as BIRCH,
- using a portfolio of very different clustering algorithms e.g. using density based, k-means or soft variants such as fuzzy c-means, graph partitioning based, statistical mechanics based, etc.

It is well known that no clustering method is perfect, and the performance of a given method can vary significantly across data-sets. For example, the popular k-means algorithm that performs respectably in several applications, has an underlying assumption that the data (after appropriate normalization such as using Mahalanobis distance) is generated by a mixture of k Gaussians, each having identical, isotropic covariance matrices. If the actual data distribution is very different from this generative model, then, even with the 'correct' guess for k, its performance can be worse than a *random* partitioning, specially in sparse, high dimensional spaces (see chapter 3). In fact, for difficult data-sets, comparative studies across multiple clustering algorithms typically show much more variability in results than comparative studies on multiple classification methods, where any good approximator of Bayesian a posteriori class probabilities will provide comparable results [RL91]. Thus there should be a potential for greater gains when using an ensemble for

clustering problems.

A different but related motivation for using a cluster ensemble is to build a *robust* clustering portfolio that can perform well over a wide range of data-sets with little hand-tuning. For example, by using an ensemble that includes approaches suitable for low-dimensional, metric spaces (e.g., k-means, SOM [Koh95], DBSCAN [EKSX96]) as well as algorithms tailored for high-dimensional, sparse spaces (e.g., spherical k-means [DM01], and Jaccard-based graph partitioning (e.g., see chapter 3), one may perform very well in 3D as well as in 30000D spaces without having to switch models. This characteristic is very attractive in practical settings.

- *Knowledge Reuse.* Another important consideration is the *reuse of existing clusterings*. In several applications, a variety of clusterings for the objects under consideration may already exist. For example, on the web, pages are categorized e.g., by Yahoo! (according to a manually-crafted taxonomy), by your Internet service provider (according to request patterns and frequencies) and by your personal bookmarks (according to your preferences). In grouping customers for a direct marketing campaign, various legacy customer segmentations might already exist, based on demographics, credit rating, geographical region, or income tax bracket. In addition, customers can be clustered based on their purchasing history (e.g., market-baskets). Can we reuse such pre-existing knowledge to create a single consolidated clustering? Knowledge reuse in this context means that we exploit the information in the provided cluster labels *without* going back to the *original features* or the algorithms

that were used to create the clusterings.

- *Distributed Computing.* The ability to deal with clustering in a distributed fashion is useful to improve scalability, security, and reliability. Also, real applications often involve distributed databases due to organizational or operational constraints [KC00]. One can argue that by transferring all data to a single warehouse and performing a series of merges and joins, one can get a single (albeit very large), flat file. A traditional algorithm can be used after randomizing and subsampling this file. But in real applications this approach may not be feasible because of the computational, bandwidth and storage costs. In certain cases, it may not even be possible for a variety of practical reasons including security, privacy, proprietary nature of data, need for fault tolerant distribution of data and services, real-time processing requirements, statutory constraints imposed by law, etc. [PCS00].

  A cluster ensemble can combine individual results from the distributed computing entities, under two scenarios:

  - *Object-distributed data.* Each processor / clusterer has access to only a subset of all objects, and can thus only cluster the observed objects. For example, corporations tend to split their customers regionally for more efficient management. Analysis such as clustering is often performed locally, and a cluster ensemble provides a way of obtaining a holistic analysis without complete integration of the local data warehouses. Note that in order to successfully combine clusterings, the subsets of objects observed by the individual clusterers must have some overlap.

– *Feature-distributed data.* In this scenario, each processor / clusterer sees only a limited number of features or attributes of each object, i.e. it observes a particular *aspect* or *view* of the data. Aspects can be completely disjoint features or have partial overlaps.

Contributions. The idea of integrating independently computed clusterings into a combined clustering leads us to a general knowledge reuse framework that we call the cluster ensemble. Our contribution in this chapter is to formally define the design of a cluster ensemble as an optimization problem and to propose three effective and efficient frameworks for solving it. We show how any set of clusterings can be represented as a hypergraph. Using this hypergraph, three methods are proposed for computing a combined clustering based on similarity-based reclustering, hypergraph partitioning, and meta-clustering, respectively. We also describe applications of cluster ensembles for each of the three application scenarios described above and show results on real data.

Notation. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ denote a set of objects / samples / points. A partitioning of these n objects into k clusters can be represented as a set of k sets of objects $\{\mathcal{C}_\ell | \ell = 1, \ldots, k\}$ or as a label vector $\lambda \in \mathbb{N}^n$. We also refer to a clustering / labeling as a model. A clusterer Φ is a function that delivers a label vector given an ordered set of objects. Figure 5.1 shows the basic setup of the cluster ensemble: A set of r labelings $\lambda^{(1,\ldots,r)}$ is combined into a single labeling λ using a consensus function Γ. We call the original set of labelings, the *profile* and the resulting single labeling, the *consensus clustering*. Vector / matrix transposition is indicated with a superscript $\dagger$. A superscript in brackets denotes an index and not an exponent.

Organization. In the next section, we will introduce the cluster en-

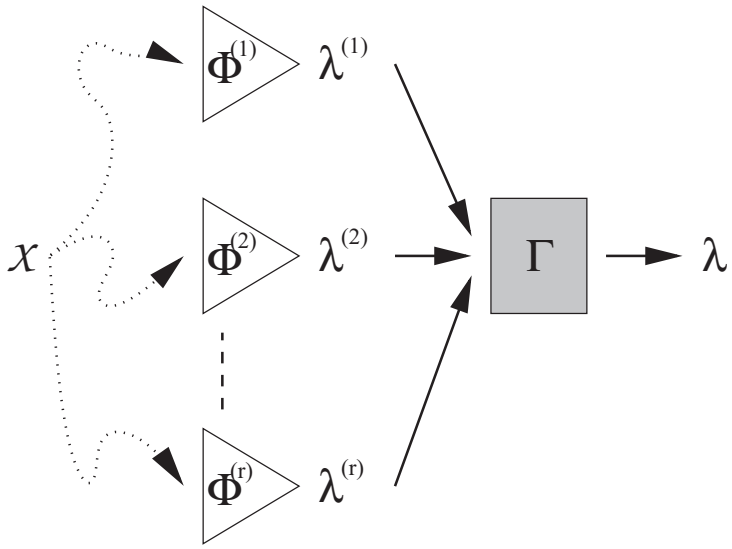


Figure 5.1: The cluster ensemble. A consensus function Γ combines clusterings $\lambda^{(q)}$ from a variety of sources, without resorting to the original object features $\mathcal{X}$ or algorithms Φ.

semble problem in more detail and in section 5.3 we propose and compare three possible combining schemes, Γ. In section 5.4 we will apply these schemes to both centralized and distributed clustering problems and present case studies on a variety of data-sets to highlight these applications.

5.2 The Cluster Ensemble Problem

5.2.1 Illustrative Example

First, we will illustrate combining of clusterings using a simple example. Consider the seven points on the 2D plane shown in figure 5.2. Four views are shown that project the data onto a 1D line. The square brackets denote the

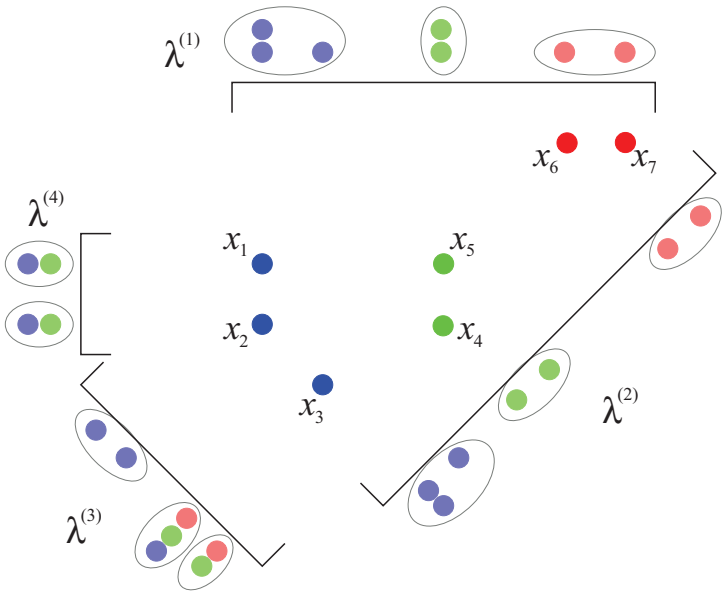


Figure 5.2: Illustration of multiple views generating different clusterings. Seven objects in the 2D plane $x_1 \ldots x_7$ are projected onto lines as illustrated by the square brackets. Only objects within the square brackets are observed. The clusters in these views (shown as bubbles) may deviate significantly from the original groups (shown by colors) and vary tremendously amongst themselves. If the original 2D and 1D features are unavailable and only the cluster labels in the views are known, the input to the cluster ensemble is the same as in table 5.1.

direction of projection as well as the regions of observation. The scenario depicted in figure 5.2 is one out of many that could have generated the following cluster ensemble problem. Please note that there is no feature information available to the cluster ensembles. The 2D spatial position of the seven data points is not know in the cluster ensemble problem (and neither are the projected 1D positions). The combining can only be based on the cluster information. Let the following label vectors specify four clusterings of the same set

of seven objects (see also table 5.1):

$$\lambda^{(1)} = (1, 1, 1, 2, 2, 3, 3)^\dagger$$
$$\lambda^{(2)} = (2, 2, 2, 3, 3, 1, 1)^\dagger$$
$$\lambda^{(3)} = (1, 1, 2, 2, 3, 3, 3)^\dagger$$
$$\lambda^{(4)} = (1, 2, ?, 1, 2, ?, ?)^\dagger$$

Inspection of the label vectors reveals that clusterings 1 and 2 are logically identical. Clustering 3 introduces some dispute about objects 3 and 5. Clustering 4 is quite inconsistent with all the other ones, has two groupings instead of 3, and also contains missing data. Now let us look for a good combined clustering with 3 clusters. Intuitively, a good combined clustering should *share as much information as possible* with the given 4 labelings. Inspection suggests that a reasonable integrated clustering is $(2, 2, 2, 3, 3, 1, 1)^\dagger$ (or one of the 6 equivalent clusterings such as $(1, 1, 1, 2, 2, 3, 3)^\dagger$). In fact, after performing an exhaustive search over all 301 unique clusterings of 7 elements into 3 groups, it can be shown that this clustering shares the maximum information with the given 4 label vectors (in terms that are more formally introduced in the next subsection).

This simple example illustrates some of the challenges. We have already seen that the label vector is not unique. In fact, for each unique clustering there are $k!$ equivalent representations as integer label vectors. However, only one representation satisfies the following two constraints: (i) $\lambda_1 = 1$; (ii) for all $i = 1, \ldots, n-1 : \lambda_{i+1} \leq \max_{j=1,\ldots,i}(\lambda_j) + 1$. By allowing only representations that fulfill both constraints, the integer vector representation can be forced to be *unique*. Transforming the labels into this 'canonical form' solves the combining problem if all clusterings are actually the same. However, if there is any discrepancy among labelings, one has to also deal with a correspondence

problem. In fact, there are $(k!)^{r-1}$ possible association patterns. In general, the number of clusters found as well as each cluster's interpretation may vary tremendously among models.

5.2.2 Objective Function for Cluster Ensembles

Given r groupings with the q-th grouping $\lambda^{(q)}$ having $k^{(q)}$ clusters, a consensus function Γ is defined as a function $\mathbb{N}^{n\times r} \to \mathbb{N}^n$ mapping a set of clusterings to an integrated clustering:

$$\Gamma : \{\lambda^{(q)} \mid q \in \{1, \ldots, r\}\} \to \lambda. \tag{5.1}$$

We abbreviate the set of groupings $\mathbf{\Lambda} = \{\lambda^{(q)} \mid q \in \{1, \ldots, r\}\}$. The optimal combined clustering should share the most information with the original clusterings. But how do we measure shared information between clusterings? In information theory, mutual information is a symmetric measure to quantify the statistical information shared between two distributions [CT91]. Suppose there are two labelings $\lambda^{(a)}$ and $\lambda^{(b)}$. Let there be $k^{(a)}$ groups in $\lambda^{(a)}$ and $k^{(b)}$ groups in $\lambda^{(b)}$. Let $n^{(h)}$ be the number of objects in cluster $\mathcal{C}_h$ according to $\lambda^{(a)}$, and n_ℓ the number of objects in cluster $\mathcal{C}_\ell$ according to $\lambda^{(b)}$. Let $n_\ell^{(h)}$ denote the number of objects that are in cluster h according to $\lambda^{(a)}$ as well as in group ℓ according to $\lambda^{(b)}$. Then, a [0,1]-normalized mutual information criterion $\phi^{(\mathrm{NMI})}$ is computed as follows (see appendix B.1 for details):

$$\phi^{(\mathrm{NMI})}(\lambda^{(a)}, \lambda^{(b)}) = \frac{2}{n} \sum_{\ell=1}^{k^{(a)}} \sum_{h=1}^{k^{(b)}} n_\ell^{(h)} \log_{k^{(a)} \cdot k^{(b)}} \left(\frac{n_\ell^{(h)} n}{n^{(h)} n_\ell} \right) \tag{5.2}$$

Therefore, the Average Normalized Mutual Information (ANMI) between a set of r labelings, $\mathbf{\Lambda}$, and a labeling $\hat{\lambda}$ is defined as follows:

$$\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \hat{\lambda}) = \frac{1}{r} \sum_{q=1}^{r} \phi^{(\mathrm{NMI})}(\hat{\lambda}, \lambda^{(q)}) \tag{5.3}$$

We define the optimal combined clustering $\lambda^{(k-\mathrm{opt})}$ to be the one that has maximal average mutual information with all individual labelings $\lambda^{(q)}$ in $\mathbf{\Lambda}$ given that the number of consensus clusters desired is k. Our objective function is average normalized mutual information (ANMI, equation 5.3), and $\lambda^{(k-\mathrm{opt})}$ is defined as

$$\lambda^{(k-\mathrm{opt})} = \arg\max_{\hat{\lambda}} \sum_{q=1}^{r} \phi^{(\mathrm{NMI})}(\hat{\lambda}, \lambda^{(q)}) \tag{5.4}$$

where $\hat{\lambda}$ goes through all possible k-partitions [SG02a].

There may be situations where not all labels are known for all objects, i.e. there is missing data in the label vectors. For such cases, the consensus clustering objective from equation 5.4 can be generalized by computing a weighted average of the mutual information with the known labels, with the weights proportional to the comprehensiveness of the labelings as measured by the fraction of known labels. Let $\mathcal{L}^{(q)}$ be the set of object indices with known labels for the q-th labeling. Then, the generalized objective function becomes:

$$\lambda^{(k-\mathrm{opt})} = \arg\max_{\hat{\lambda}} \sum_{q=1}^{r} |\mathcal{L}^{(q)}| \phi^{(\mathrm{NMI})}(\hat{\lambda}_{\mathcal{L}^{(q)}}, \hat{\lambda}^{(q)}_{\mathcal{L}^{(q)}}) \tag{5.5}$$

For finite populations, the trivial solution is to exhaustively search through all possible clusterings with k labels for the one with the maximum ANMI. However, for n objects and k partitions there are

$$\frac{1}{k!} \sum_{\ell=1}^{k} \begin{pmatrix} k \\ \ell \end{pmatrix} (-1)^{k-\ell} \ell^n \tag{5.6}$$

possible clusterings, or approximately $k^n/k!$ for $n \gg k$. For example, there are 171,798,901 ways to form 4 groups of 16 objects. For 8 groups of 32 objects this number grows to over $1.75 \cdot 10^{24}$ and for 16 groups of 64 objects to over $4.23 \cdot 10^{63}$. Clearly, this exponential growth in n makes a full search prohibitive. Having defined the objective, we now propose three algorithms to find good heuristic solutions. Note that greedy optimizations of the ANMI objective are difficult since it is a hard combinatorial problem. So our three proposed algorithms in this chapter have been developed from intuitive ideas for maximizing the objective.

5.3 Efficient Consensus Functions

In this section, we introduce three efficient heuristics to solve the cluster ensemble problem. All algorithms approach the problem by first transforming the set of clusterings into a hypergraph representation.

Cluster-based Similarity Partitioning Algorithm (CSPA). A clustering signifies a relationship between objects in the same cluster and can thus be used to establish a measure of pairwise similarity. This *induced similarity measure* is then used to recluster the objects, yielding a combined clustering.

HyperGraph Partitioning Algorithm (HGPA). In this algorithm, we approximate the maximum mutual information objective with a constrained *minimum cut* objective. Essentially, the cluster ensemble problem is posed as a partitioning problem of a suitably defined hypergraph where hyperedges represent clusters.

Meta-CLustering Algorithm (MCLA). Here, the objective of integration is viewed as a *cluster correspondence problem.* Essentially, groups of clusters (meta-clusters) have to be identified and consolidated.

The following four subsections describe the common hypergraph representation, CSPA, HGPA, and MCLA. Section 5.3.5 discusses differences in the algorithms and evaluates their performance in a controlled experiment.

5.3.1 Representing Sets of Clusterings as a Hypergraph

The first step for both of our proposed consensus functions is to transform the given cluster label vectors into a suitable hypergraph representation. In this subsection, we describe how any set of clusterings can be mapped to a hypergraph. A hypergraph consists of vertices and hyperedges. An edge in a regular graph connects exactly 2 vertices. A hyperedge is a generalization of an edge in that it can connect any *set* of vertices.

For each label vector $\lambda^{(q)} \in \mathbb{N}^n$, we construct the binary membership indicator matrix $\mathbf{H}^{(q)} \in \mathbb{N}^{n \times k^{(q)}}$ in which each cluster is represented as a hyperedge (column), as illustrated in table 5.1.[3] All entries of a row in the binary membership indicator matrix $\mathbf{H}^{(q)}$ add to 1, if the row corresponds to an object with *known* label. Rows for objects with unknown label are all zero.

The concatenated block matrix $\mathbf{H} = \mathbf{H}^{(1,\ldots,r)} = (\mathbf{H}^{(1)} \ \ldots \ \mathbf{H}^{(r)})$ defines the adjacency matrix of a hypergraph with n vertices and $\sum_{q=1}^{r} k^{(q)}$ hyperedges. Each column vector $\mathbf{h}_a$ specifies a hyperedge h_a, where 1 indicates that the vertex corresponding to the row is part of that hyperedge and 0 indicates

[3]When generalizing our algorithms to *soft* clustering, $\mathbf{H}^{(q)}$ simply contains the posterior probabilities of cluster membership.

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$	$\lambda^{(4)}$			$\mathbf{H}^{(1)}$			$\mathbf{H}^{(2)}$			$\mathbf{H}^{(3)}$			$\mathbf{H}^{(4)}$	
							$\mathbf{h}_1$	$\mathbf{h}_2$	$\mathbf{h}_3$	$\mathbf{h}_4$	$\mathbf{h}_5$	$\mathbf{h}_6$	$\mathbf{h}_7$	$\mathbf{h}_8$	$\mathbf{h}_9$	$\mathbf{h}_{10}$	$\mathbf{h}_{11}$
x_1	1	2	1	1		v_1	1	0	0	0	1	0	1	0	0	1	0
x_2	1	2	1	2		v_2	1	0	0	0	1	0	1	0	0	0	1
x_3	1	2	2	?	$\Leftrightarrow$	v_3	1	0	0	0	1	0	0	1	0	0	0
x_4	2	3	2	1		v_4	0	1	0	0	0	1	0	1	0	1	0
x_5	2	3	3	2		v_5	0	1	0	0	0	1	0	0	1	0	1
x_6	3	1	3	?		v_6	0	0	1	1	0	0	0	0	1	0	0
x_7	3	1	3	?		v_7	0	0	1	1	0	0	0	0	1	0	0

Table 5.1: Illustrative cluster ensemble problem with $r = 4$, $k^{(1,\ldots,3)} = 3$, and $k^{(4)} = 2$: Original label vectors (left) and equivalent hypergraph representation with 11 hyperedges (right). Each cluster is transformed into a hyperedge.

that it is not. Thus, we have mapped each cluster to a hyperedge and the set of clusterings to a hypergraph.

5.3.2 Cluster-based Similarity Partitioning Algorithm (CSPA)

Essentially, if two objects are in the same cluster then they are considered to be fully similar, and if not they are dissimilar. This is the simplest heuristic and is used in the Cluster-based Similarity Partitioning Algorithm (CSPA). With this viewpoint, one can simply reverse engineer a single clustering into a binary similarity matrix. Similarity between two objects is 1 if they are in the same cluster and 0 otherwise. For each clustering, a $n \times n$ binary similarity matrix is created. The entry-wise average of r such matrices representing the r sets of groupings yields an overall similarity matrix. Figure 5.3 illustrates the generation of the cluster-based similarity matrix for the example given in table 5.1.

Alternatively, and more concisely, this can be interpreted as using k binary cluster membership features and defining similarity as the fraction of clusterings in which two objects are in the same cluster. The entire $n \times n$ similarity matrix $\mathbf{S}$ can be computed in one sparse matrix multiplication $\mathbf{S} = \frac{1}{r}\mathbf{H}\mathbf{H}^{\dagger}$.[4]

Now, we can use the similarity matrix to recluster the objects using any reasonable similarity based clustering algorithm. We chose to partition the induced similarity graph (vertex = object, edge weight = similarity) using METIS [KK98a] because of its robust and scalable properties.

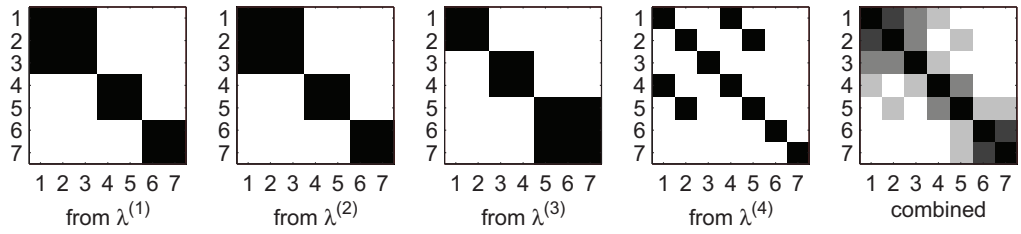


Figure 5.3: Illustration of Cluster-based Similarity Partitioning Algorithm (CSPA) for the cluster ensemble example problem given in table 5.1. Each clustering contributes a similarity matrix (matrix entries are shown by darkness proportional to similarity). Their average is then used to recluster the objects to yield consensus.

5.3.3 HyperGraph Partitioning Algorithm (HGPA)

The second algorithm is a direct approach to cluster ensembles that re-partitions the data using the given clusters as indications of strong bonds. The cluster

[4]This approach can be extended to soft clusterings through using the objects' posterior probabilities cluster of cluster membership in $\mathbf{H}$.

ensemble problem is formulated as *partitioning* the hypergraph by cutting a *minimal* number of hyperedges. We call this approach the HyperGraph Partitioning Algorithm (HGPA). All hyperedges are considered to have the same weight. Also, all vertices are equally weighted. Note, that this includes n_ℓ-way relationship information, while CSPA only considers pairwise relationships. Now, we look for a hyperedge separator that partitions the hypergraph (figure 5.4) into k unconnected components of approximately the same size. Equal sizes are obtained by maintaining a vertex imbalance of at most 5% as formulated by the following constraint: $k \cdot \max_{\ell \in \{1,\ldots,k\}} \frac{n_\ell}{n} \leq 1.05$.

Hypergraph partitioning is a well-studied area [KL70, AK95] and algorithmic details are omitted here for brevity. We use the hypergraph partitioning package HMETIS [KAKS97]. HMETIS gives high-quality partitions and is very scalable. However, please note that hypergraph partitioning in general has no provision for partially cut hyperedges. This means that there is no sensitivity to how much of a hyperedge is left in the same group after the cut. This can be problematic for our applications. Let us consider the example from table 5.1. For simplicity, let us assume that only the three hyperedges from $\lambda^{(1)}$ are present. The two partitionings $\{\{x_1, x_2, x_7\}, \{x_3, x_4\}, \{x_5, x_6\}\}$ and $\{\{x_1, x_7\}, \{x_3, x_4\}, \{x_2, x_5, x_6\}\}$ both cut all three hyperedges. The first is intuitively superior, because 2/3 of the hyperedge $\{x_1, x_2, x_3\}$ 'remain' versus only 1/3 in the second. However, in standard hypergraph partitioning they have equivalent quality since both cut the same number of hyperedges.

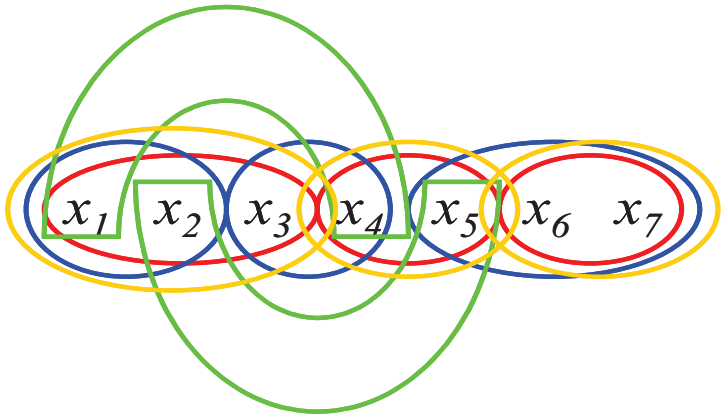


Figure 5.4: Illustration of HyperGraph Partitioning Algorithm (HGPA) for the cluster ensemble example problem given in table 5.1. Each hyperedge is represented by a closed curve enclosing the vertices it connects. Hyperedges from the same clustering have the same color. The combined clustering $\{\{x_1, x_2, x_3\}, \{x_4, x_5\}, \{x_6, x_7\}\}$ has the minimal hyperedge cut of 4 and is as balanced as possible for 3 clusters of 7 objects.

5.3.4 Meta-CLustering Algorithm (MCLA)

In this subsection, we introduce the third algorithm to solve the cluster ensemble problem. The Meta-CLustering Algorithm (MCLA) is based on clustering clusters. It also yields object-wise confidence estimates of cluster membership.

We represented each cluster by a hyperedge. The idea in MCLA is to group and collapse related hyperedges and assign each object to the collapsed hyperedge in which it participates most strongly. The hyperedges that are considered related for the purpose of collapsing are determined by a graph-based clustering of hyperedges. We refer to each cluster of hyperedges as a meta-cluster $\mathcal{C}^{(\mathrm{M})}$. Collapsing reduces the number of hyperedges from $\sum_{q=1}^{r} k^{(q)}$ to k. The detailed steps are:

Construct Meta-graph. Let us view all the $\sum_{q=1}^{r} k^{(q)}$ indicator vectors $\mathbf{h}$ (the hyperedges of $\mathbf{H}$) as vertices of another regular undirected graph, the meta-graph. The edge weights are proportional to the similarity between vertices. A suitable similarity measure here is the binary Jaccard measure, since it is the ratio of the intersection to the union of the sets of objects corresponding to the two hyperedges. Formally, the edge weight $w_{a,b}$ between two vertices h_a and h_b as defined by the binary Jaccard measure of the corresponding indicator vectors $\mathbf{h}_a$ and $\mathbf{h}_b$ is: $w_{a,b} = \frac{\mathbf{h}_a^\dagger \mathbf{h}_b}{\|\mathbf{h}_a\|_2^2 + \|\mathbf{h}_b\|_2^2 - \mathbf{h}_a^\dagger \mathbf{h}_b}$.

Since the clusters are non-overlapping (e.g., hard), there are no edges amongst vertices of the same clustering $\mathbf{H}^{(q)}$ and, thus, the meta-graph is r-partite, as shown in figure 5.5.

Cluster Hyperedges. Find matching labels by partitioning the meta-graph into k balanced meta-clusters. Each vertex is weighted proportional to the size of the corresponding cluster. Balancing ensures that the sum of vertex-weights is approximately the same in each meta-cluster. We use the graph partitioning package METIS in this step. This results in a clustering of the $\mathbf{h}$ vectors. Since each vertex in the meta-graph represents a distinct cluster label, a meta-cluster represents a group of corresponding labels.

Collapse Meta-clusters. For each of the k meta-clusters, we collapse the hyperedges into a single meta-hyperedge. Each meta-hyperedge has an association vector which contains an entry for each object describing its level of association with the corresponding meta-cluster. The level is computed by averaging all indicator vectors $\mathbf{h}$ of a particular meta-

cluster.[5] An entry of 0 or 1 indicates the weakest or strongest association, respectively.

Compete for Objects. In this step, each object is assigned to its most associated meta-cluster: Specifically, an object is assigned to the meta-cluster with the highest entry in the association vector. Ties are broken randomly. The confidence of an assignment is reflected by the winner's share of association (ratio of the winner's association to the sum of all other associations). Note that not every meta-cluster can be guaranteed to win at least one object. Thus, there are *at most* k labels in the final combined clustering λ.

Figure 5.5 illustrates meta-clustering for the example given in table 5.1 where $r = 4$, $k = 3$, $k^{(1,\ldots,3)} = 3$, and $k^{(4)} = 2$. Figure 5.5 shows the original 4-partite meta-graph. The three meta-clusters are shown in red / $\circ$, blue / $\times$, and green / $+$. Consider the first meta-cluster, $\mathcal{C}_1^{(\mathrm{M})} = \{h_3, h_4, h_9\}$ (the red/$\circ$ markers in figure 5.5). Collapsing the hyperedges yields the object-weighted meta-hyperedge $h_1^{(\mathrm{M})} = \{v_5, v_6, v_7\}$ with association vector $(0, 0, 0, 0, 1/3, 1, 1)^\dagger$. Subsequently, meta-cluster $\mathcal{C}_1^{(\mathrm{M})}$ will win the competition for vertices/objects v_6 and v_7, and thus represent the cluster $\mathcal{C}_1 = \{x_6, x_7\}$ in the resulting integrated clustering. Our proposed meta-clustering algorithm robustly outputs $(2, 2, 2, 3, 3, 1, 1)^\dagger$, one of the 6 optimal clusterings which is equivalent to clusterings $\lambda^{(1)}$ and $\lambda^{(2)}$. The uncertainty about some objects is reflected in the confidences 3/4, 1, 2/3, 1, 1/2, 1, and 1 for objects 1 through 7, respectively.

[5] A weighted average can be used if the initial clusterings have associated confidences as in soft clustering.

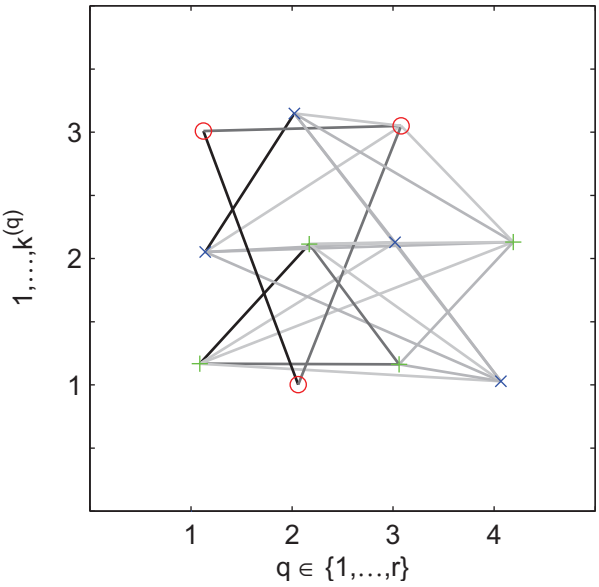


Figure 5.5: Illustration of Meta-CLustering Algorithm (MCLA) for the cluster ensemble example problem given in table 5.1. The 4-partite meta-graph is shown. Edge darkness increases with edge weight. The vertex positions are slightly perturbed to expose otherwise occluded edges. The three meta-clusters are shown in red / $\circ$, blue / $\times$, and green / $+$.

5.3.5 Discussion and Comparison

Let us first take a look at the worst case time complexity of the proposed algorithms. Assuming quasi-linear (hyper-)graph partitioners such as (H)METIS, CSPA is $O(n^2kr)$, HGPA is $O(nkr)$, and MCLA is $O(nk^2r^2)$. The fastest is HGPA, closely followed by MCLA since k tends to be small. CSPA is the slowest and can become intractable for large n.

We performed a controlled experiment that allows us to compare the properties of the three proposed consensus functions. First, a we partition $n = 500$ objects into $k = 10$ groups at random to obtain the original clus-

tering κ.[6] We duplicate this clustering $r = 10$ times. Now in each of the 10 labelings, a fraction of the labels is replaced with random labels from a uniform distribution from 1 to k. Then, we feed the noisy labelings to the proposed consensus functions. The resulting combined labeling is evaluated in two ways. Firstly, we measure the normalized objective function $\phi^{(\mathrm{ANMI})}(\boldsymbol{\Lambda}, \lambda)$ of the ensemble output λ with all the individual labels in $\boldsymbol{\Lambda}$. Secondly, we measure the normalized mutual information of each consensus labeling with the original undistorted labeling using $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$. For better comparison, we added a random label generator as a baseline method. Also, performance measures of a hypothetical consensus function that returns the original labels are included to illustrate maximum performance for low noise settings.[7]

Figure 5.6 shows the results. As noise increases, labelings share less information and thus maximum obtainable $\phi^{(\mathrm{ANMI})}(\boldsymbol{\Lambda}, \lambda)$ decreases, and so does $\phi^{(\mathrm{ANMI})}(\boldsymbol{\Lambda}, \lambda)$ for all techniques. HGPA performs the worst in this experiment, which we believe is due to the lacking provision of partially cut edges. MCLA retains more $\phi^{(\mathrm{ANMI})}(\boldsymbol{\Lambda}, \lambda)$ than CSPA in presence of low to medium-high noise. Interestingly, in very high noise settings CSPA exceeds MCLA's performance. Note also that for such high noise settings the original labels have a lower average normalized mutual information $\phi^{(\mathrm{ANMI})}(\boldsymbol{\Lambda}, \lambda)$. This is because the set of labels are almost completely random and the consensus algorithms recover whatever little common information is present whereas the original labeling is now almost fully unrelated. However, in most cases noise should not exceed 50% and MCLA seems to perform best in this simple con-

[6]Labels were drawn from a uniform distribution from 1 to k. Therefore, groups are approximately balanced.

[7]In low noise settings, the original labels are the global maximum, since they share the most mutual information with the distorted labelings.

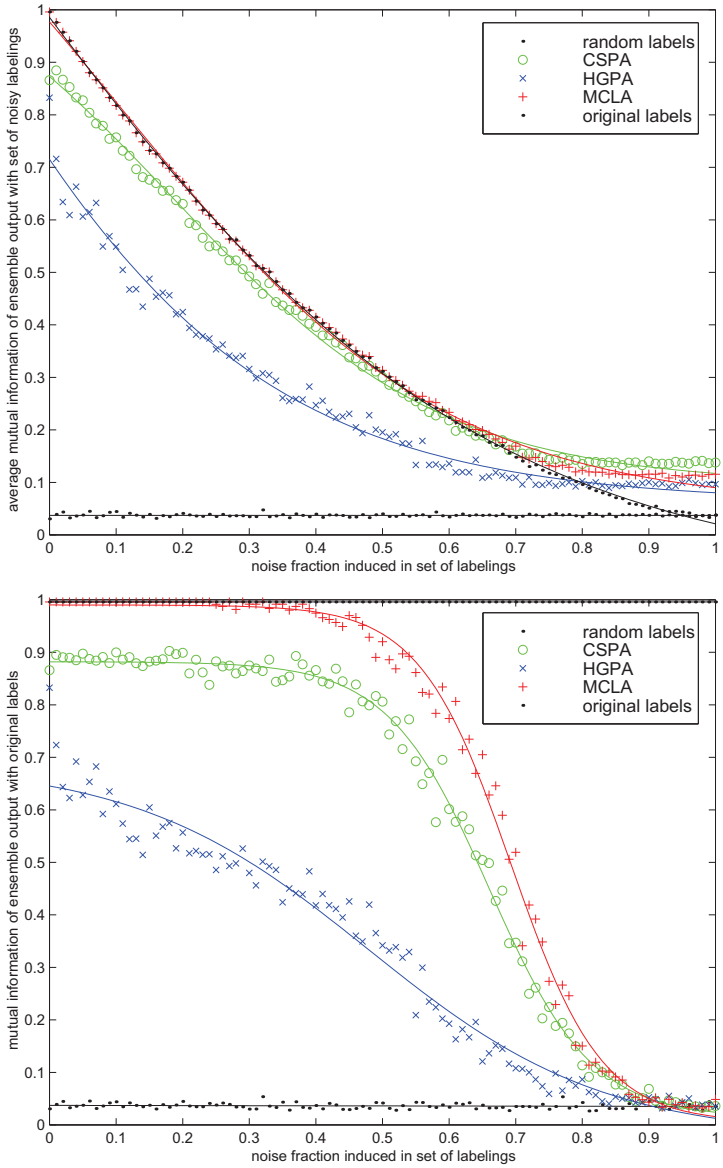


Figure 5.6: Comparison of consensus functions in terms of the objective function $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ (top) and in terms of their normalized mutual information with original labels $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ (bottom) for various noise levels. A fitted sigmoid (least squared error) is shown for all algorithms to show the trend.

trolled experiment. Figure 5.6 illustrates how well the algorithms can recover the true labeling in the presence of noise for robust clustering. As noise increases labelings share less information with the true labeling and thus the ensemble's $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ decreases. The ranking of the algorithms is the same using this measure with MCLA best, followed by CSPA, and HGPA worst. In fact, MCLA recovers the original labeling at up to 35% noise in this scenario! For less than 50%, the algorithms have the same ranking regardless of whether $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ or $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ is used. This indicates that our proposed objective function $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ is indeed appropriate since in real applications, κ and, thus, $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ is not available.

This experiment indicates that MCLA should be best suited in terms of time complexity as well as quality. In the applications and experiments described in the following sections we observe that each combining method can result in a higher ANMI than the others for particular setups. In fact, we found that MCLA tends to be best in low noise/diversity settings and HGPA/CSPA tend to be better in high noise/diversity settings. This is because MCLA assumes that there are meaningful cluster correspondences which is more likely to be true when there is little noise and less diversity. Thus, it is useful to have all three methods.

Indeed, our objective function has an added advantage that it allows one to add a stage that selects the best consensus function without any supervision information, by simply selecting the one with the highest ANMI. So, for the experiments in this chapter, we first report the results of this 'supra'-consensus function Γ, obtained by running *all three* algorithms, CSPA, HGPA and MCLA, and selecting the one with the greatest ANMI. Then, if there are significant differences or notable trends observed among the three algorithms,

this further level of detail is described. Note that the supra-consensus function is completely unsupervised and avoids the problem of selecting the best combiner for a data-set beforehand.

5.4 Cluster Ensemble Applications and Experiments

Consensus functions *enable* a variety of new approaches to several problems. After introducing the data-sets used and the evaluation methodology, in subsection 5.4.3 we illustrate how robustness of clustering can be increased through combining a set of clusterings. In subsection 5.4.4, we discuss how distributed clustering can be performed when each entity only has access to a very limited subset of features. Finally, subsection 5.4.5 shows how one can operate on several very limited subset of objects and combine them afterwards thereby enabling distributed clustering.

5.4.1 Data-sets

We illustrate the cluster ensemble applications on two real and two artificial data-sets. In table 5.2 some basic properties of the data-sets (left) and parameter choices (right) are summarized. The first data-set (`2D2K`) was artificially generated and contains 500 points each of two 2-dimensional (2D) Gaussian clusters with means $(-0.227, 0.077)^\dagger$ and $(0.095, 0.323)^\dagger$ and equal variance of 0.1. The second data-set (`8D5K`) contains 1000 points from 5 multivariate Gaussian distributions (200 points each) in 8D space. Again, clusters all have the same variance (0.1), but different means. Means were drawn from a uniform

distribution within the unit hypercube. Both artificial data-sets are illustrated in appendix A.1 and are available for download at `http://strehl.com/`.

The third data-set (`PENDIG`, appendix A.3) is for pen-based recognition of handwritten digits. It is publicly available from the UCI Machine Learning Repository and was contributed by Alpaydin and Alimoglu. It contains 16 spatial features for each of the 7494 training and 3498 test cases (objects). There are ten classes of roughly equal size (balanced clusters) in the data corresponding to the digits 0 to 9.

The fourth data-set is for text clustering. The 20 original Yahoo! news categories in the data are `Business`, `Entertainment` (`no sub-category`, `art`, `cable`, `culture`, `film`, `industry`, `media`, `multimedia`, `music`, `online`, `people`, `review`, `stage`, `television`, `variety`), `Health`, `Politics`, `Sports`, `Technology`. The data is publicly available from `ftp://ftp.cs.umn.edu/ /dept/users/boley/` (K1 series) and was used in [BGG+99, SGM00]. The raw 21839 $\times$ 2340 word-document matrix consists of the non-normalized occurrence frequencies of stemmed words, using Porter's suffix stripping algorithm [FBY92]. Pruning all words that occur less than 0.01 or more than 0.10 times on average because they are insignificant (e.g., `haruspex`) or too generic (e.g., `new`), respectively, results in $d = 2903$. We call this data-set `YAHOO` (see also appendix A.5).

For `2D2K`, `8D5K`, and `PENDIG` we use $k = 2$, 5, and 10, respectively. When clustering `YAHOO`, we use $k = 40$ clusters unless noted otherwise. We chose two times the number of categories, since this seemed to be the more natural number of clusters as indicated by preliminary runs and visualization.[8]

[8]Using a greater number of clusters than categories can be viewed as allowing multi-modal distributions for each category. For example, in a noisy XOR problem, there are two categories, but four clusters.

For 2D2K, 8D5K, and PENDIG we use Euclidean-based similarity. For YAHOO we use cosine-based similarity.

name	features	#features	#categories	balance	similarity	#clusters
2D2K	real	2	2	1.00	Euclidean	2
8D5K	real	8	5	1.00	Euclidean	5
PENDIG	real	16	10	0.87	Euclidean	10
YAHOO	ordinal	2903	20	0.24	Cosine	40

Table 5.2: Overview of data-set properties and parameters for cluster ensemble experiments. Balance is defined as the ratio of the average category size to the largest category size.

5.4.2 Evaluation Criterion

Evaluation of the quality of a clustering is a non-trivial and often ill-posed task. In fact, many definitions of objective functions for clusterings exist [JD88]. *Internal* criteria formulate quality as a function of the given data and/or similarities. For example, the mean squared error criterion (for k-means) and other measures of compactness are popular evaluation criteria. Measures can also be based on isolation such as the min-cut criterion, which uses the sum of edge weights across clusters (for graph partitioning). When using internal criteria, clustering becomes an optimization problem, and a clusterer can evaluate its own performance and tune its results accordingly.

External criteria on the other hand impose quality by additional, external information not given to the clusterer, such as category labels. This is sometimes more appropriate since groupings are ultimately evaluated externally by humans. For example, when objects have already been categorized by

an external source, i.e. when class labels are available, we can use information theoretic measures to quantify the match between the categorization and the clustering. Previously, average purity and entropy-based measures to assess clustering quality from 0 (worst) to 1 (best) have been used [BGG+99]. While average purity is intuitive to understand, it is biased to favor small clusters. Singletons, in fact, are scored as perfect. Also, a monolithic clustering (one single cluster for all objects) receives a score as high as the maximum category prior probability. In unstratified data-sets, this prior might be close to 1, while the desired value should be close to 0. An entropy-based measure is better than purity but is still biased towards small clusters (e.g., a set of singletons is always considered perfect).

Using a measure based on normalized mutual information (equation 5.2) fixes both biases (see chapter 4): Monolithic clusterings are evaluated at 0 and singletons are severely discounted compared to the best clustering.[9] However, the mutual information for the best clustering is smaller than 1 unless all categories have equal prior probabilities. Thus, mutual information provides an unbiased and conservative measure as compared to purity and entropy. Given g categories (or classes), we use the categorization labels κ to evaluate the cluster quality using mutual information $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$, as defined in equation 5.2.

5.4.3 Robust Centralized Clustering (RCC)

A consensus function can introduce redundancy and foster robustness when, instead of choosing or fine-tuning a single clusterer, an ensemble of clusterers

[9] For the purpose of this discussion, the best clustering is equivalent to the categorization.

is employed and their results are combined. This is particularly useful when clustering has to be performed in a closed loop without human interaction. The goal of Robust Centralized Clustering (RCC) is to do well for a wide variety of data distributions with a *fixed* ensemble of clusterers.

In RCC, each clusterer has access to all features and to all objects. However, each clusterer might take a different approach. In fact, approaches *should* be very diverse for best results. They can use different distance/similarity measures (e.g., Euclidean, cosine) or techniques (graph-based, agglomerative, k-means) (see chapter 4). The ensemble's clusterings are then integrated using the consensus function Γ without access to the original features.

To show that RCC can yield robust results in low-dimensional metric spaces as well as in high-dimensional sparse spaces *without* any modifications, the following experiment was set up. First, 10 diverse clustering algorithms were implemented: (1) self-organizing map; (2) hypergraph partitioning; k-means with distance based on (3) Euclidean, (4) cosine, (5) correlation, and (6) extended Jaccard; and graph partitioning with similarity based on (7) Euclidean, (8) cosine, (9) correlation, and (10) extended Jaccard. Implementational details of the individual algorithms can be found in chapter 4.

RCC was performed 10 times each on sample sizes of 50, 100, 200, 400, and 800, for each data-set. Different sample sizes provide insight how cluster quality improves as more data becomes available. Quality improvement depends on the clusterer as well as the data. For example, more complex data-sets require more data until quality maxes out. We also computed a random clustering for each experiment to establish a baseline performance. The quality in terms of difference in mutual information as compared to the random clustering algorithm for all 11 approaches (10 + consensus) is shown

in figure 5.7. Figure 5.8 shows learning curves for the average quality of the 10 algorithms versus RCC.

In figure 5.7(top row) the results for the 2D2K data using $k = 2$ are shown. From an external viewpoint, the consensus function was given seven good (Euclidean, cosine, and extended Jaccard based k-means as well as graph partitioning, and self-organizing feature-map) and three poor (hypergraph partitioning, correlation based k-means, and correlation based graph partitioning) clusterings. At sample size of $n = 800$, the RCC results are better than all individual algorithm quality evaluations. There is no noticeable deterioration caused by the poor clusterings. The average RCC quality at 0.85 is 48% higher than the average quality of all individual algorithms (excluding random) at 0.57.

In case of the YAHOO data (figure 5.7(bottom row)) the consensus function received three poor clusterings (Euclidean based k-means as well as graph partitioning; and self-organizing feature-map[10]), four good (hypergraph partitioning, cosine, correlation, and extended Jaccard based k-means) and three excellent (cosine, correlation, and extended Jaccard based graph partitioning) clusterings. The RCC results are almost as good as the average of the excellent clusterers despite the presence of distractive clusterings. In fact, at the $n = 800$ level, RCC's average quality of 0.38 is 19% better than the average qualities of all the other algorithms (excluding random) at 0.32. This shows that for this scenario, too, cluster ensembles work well and also are robust!

Similar results are obtained for 8D5K and PENDIG. In these two cases all individual approaches work comparably well except for hypergraph partition-

[10]In fact, the self-organizing feature-map yielded sub-random quality. This is due to the fact that it produced incoherent *as well as* imbalanced clusters.

ing. The supra-consensus function learns to ignore hypergraph partitioning results and yields a consensus clustering of good quality.

Figure 5.8 shows how RCC is consistently better in all four scenarios than picking a random / average single technique. Looking at the three consensus techniques, the need for all of them becomes apparent since there is no ubiquitous winner. In `2D2K`, `8D5K`, and `PENDIG`, MCLA generally had the highest ANMI, followed by CSPA, while HGPA performed poorly. In `YAHOO`, both CSPA and HGPA, had the highest ANMI approximately equally often, while MCLA performed poorly. We believe this is due to the fact that there was higher diversity in `YAHOO` clusterings and CSPA and HGPA are better suited for that because no cluster correspondence is assumed.

The experimental results clearly show that cluster ensembles can be used to increase robustness in risk-intolerant settings. Especially, since it is generally hard to evaluate clusters in high-dimensional problems, a cluster ensemble can be used to 'throw' many models at a problem and then integrate them using an consensus function to yield stable results. Thereby the user does not have to have, e.g., category labels to pick a single best model. Rather the ensemble automatically 'focuses' on whatever is most appropriate for the given data. In our experiments, there is diversity as well as some poorly performing clusterers. If there are diverse but comparably performing clusterers, the quality actually significantly outperforms the best individual clusterer, as we will see in the next section.

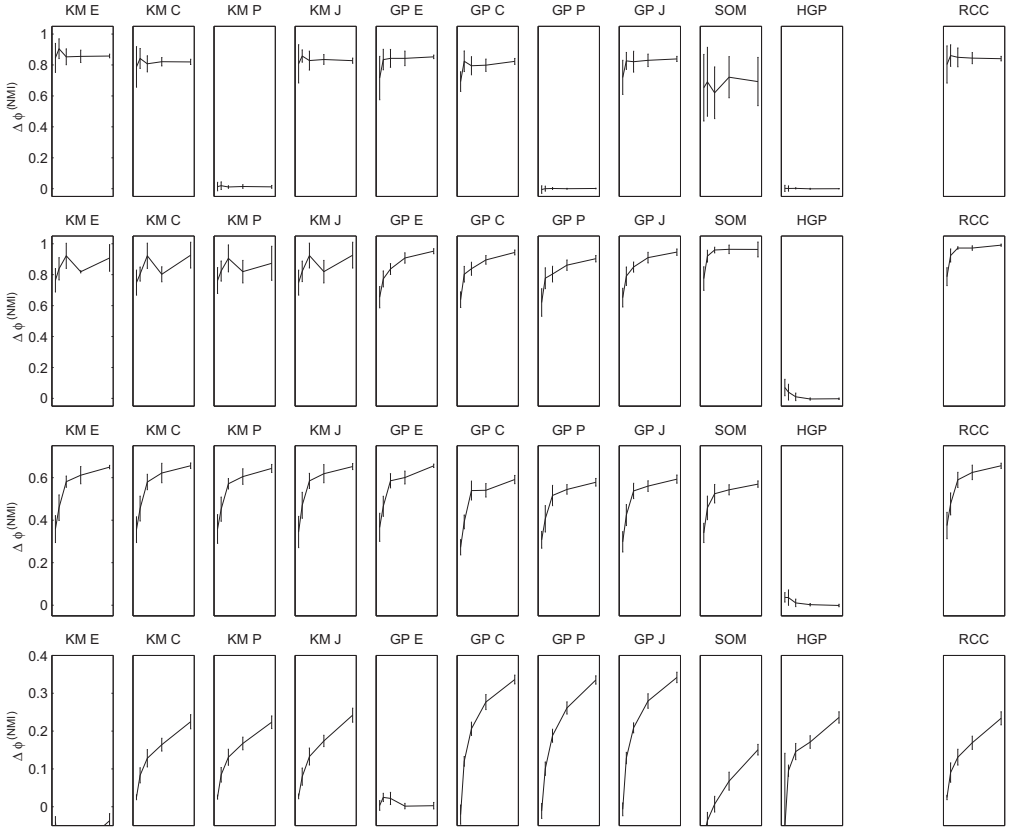


Figure 5.7: Detailed Robust Consensus Clustering (RCC) results. Learning curves for `2D2K` (top row), `8D5K` (second row), `PENDIG` (third row), and `YAHOO` (bottom row) data. Each learning curve shows the difference in mutual information-based quality $\phi^{(\mathrm{NMI})}$ compared to random for 5 sample sizes (at 50, 100, 200, 400, and 800). The bars for each data-point indicate ± 1 standard deviations over 10 experiments. Each column corresponds to a particular clustering algorithm. The rightmost column gives RCC quality for combining results of all other 10 algorithms. RCC yields robust results in all four scenarios.

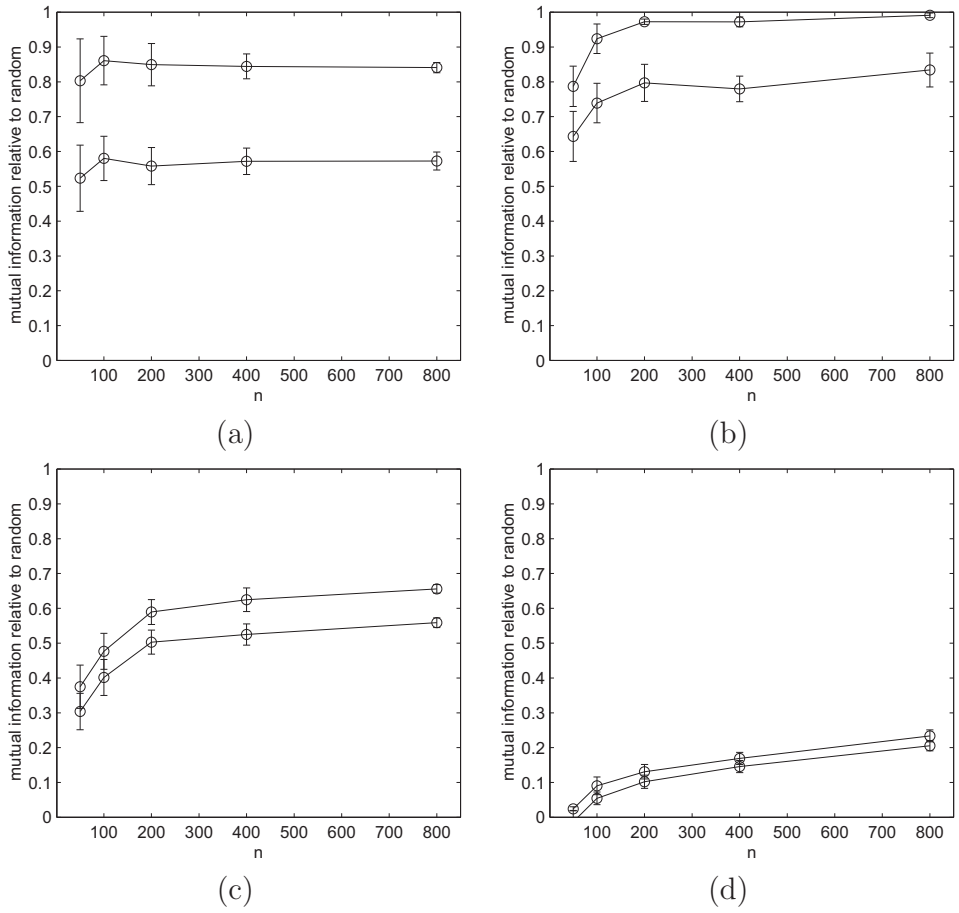


Figure 5.8: Summary of RCC results. Average learning curves and RCC learning curves for `2D2K` (a), `8D5K` (b), `PENDIG` (c), and `YAHOO` (d) data. Each learning curve shows the difference in mutual information-based quality $\phi^{(\mathrm{NMI})}$ compared to random for 5 sample sizes (at 50, 100, 200, 400, and 800). The bars for each data-point indicate ± 1 standard deviations over 10 experiments. The upper curve gives RCC quality for combining results of all other 10 algorithms. The lower curve is the average performance of the 10 algorithms. RCC yields robust results in any scenario.

5.4.4 Feature-Distributed Clustering (FDC)

In Feature-Distributed Clustering (FDC), we show how cluster ensembles can be used to boost quality of results by combining a set of clusterings obtained from partial views of the data. As mentioned in the introduction, such scenarios result in distributed databases and federated systems that cannot be pooled into one big flat file because of proprietary data aspects, privacy concerns, performance issues, etc. In such situations, it is more realistic to have one clusterer for each database and transmit only the cluster labels (but not the attributes of each record) to a central location where they can be combined using the supra-consensus function.

For our experiments, we simulate such a scenario by running several clusterers, each having access to only a restricted, small subset of features (subspace). Each clusterer has a partial *view* of the data. Each clusterer has access to all objects. The clusterers find groups in their views/subspaces using the same clustering technique. In the combining stage, individual results are integrated using our supra-consensus function to recover the full structure of the data. As this is a knowledge-reuse framework, the ensemble has no access to the original features.

First, let us discuss experimental results for the `8D5K` data, since they lend themselves well to illustration. We create 5 random 2D views (through selection of a pair of features) of the 8D data, and use Euclidean-based graph partitioning with $k = 5$ in each view to obtain 5 individual clusterings. The 5 individual clusterings are then combined using our supra-consensus function proposed in the previous section. The clusters are linearly separable in the full 8D space and clustering in the 8D space yields the original generative la-

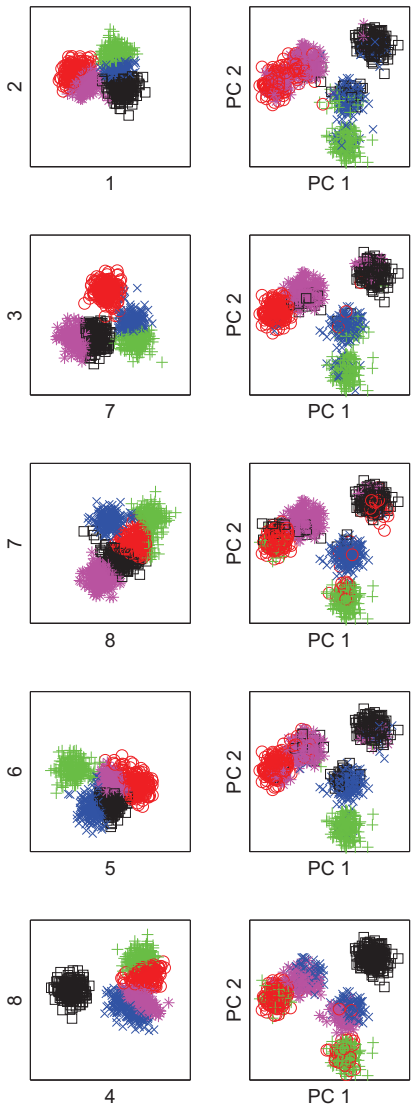
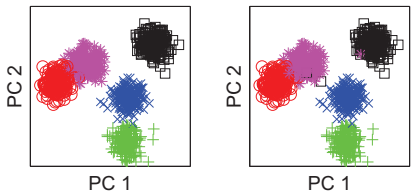


Figure 5.9: Illustration of Feature-Distributed Clustering (FDC) on `8D5K` data. Each row corresponds to a random selection of 2 out of 8 feature dimensions. For each of the 5 chosen feature pairs, a row shows the clustering (colored) obtained on the 2D subspace spanned by the selected feature pair (left) and visualizes these clusters on the plane of the global 2 principal components (right).



(a) (b)

Figure 5.10: Reference clustering (a) and FDC consensus clustering (b) of `8D5K` data. Data points are projected along the first 2 principal components of the 8D data. The reference clustering is obtained by graph partitioning using Euclidean similarity in original 8D space and is identical to the generating distribution assignment. The consensus clustering is derived from the combination of 5 clusterings, each obtained in 2D (from random feature pairs, see figure 5.9). The consensus clustering (b) is clearly superior compared to any of the 5 individual results (figure 5.9(right)) and is almost flawless compared to the reference clustering (a).

bels and is referred to as the reference clustering. Using PCA to project the data into 2D separates all 5 clusters fairly well (figure 5.10). In figure 5.10(a), the reference clustering is illustrated by coloring the data points in the space spanned by the first and second principal components (PCs). Figure 5.10(b) shows the final FDC result after combining 5 subspace clusterings. Each clustering has been computed from random feature pairs. These subspaces are shown in figure 5.9. Each of the rows corresponds to a random selection of 2 out of 8 feature dimensions. For each of the 5 chosen feature pairs, a row shows the 2D clustering (left, feature pair shown on axis) and the same 2D clustering labels used on the data projected onto the global 2 principal components (right). For consistent appearance of clusters across rows, the dot colors / shapes have been matched using meta-clusters. All points in clusters of the same meta-cluster share the same color / shape amongst all plots. In

any subspace, the clusters can not be segregated well due to strong overlaps. The supra-consensus function can combine the partial knowledge of the 5 clusterings into a far superior clustering. FDC (figure 5.10(b)) are clearly superior compared to any of the 5 individual results (figure 5.9(right)) and is almost flawless compared to the reference clustering (figure 5.10(a)). The best individual result has 120 'mislabeled' points while the consensus only has 3 points 'mislabeled'.

input and parameters			quality					
data	sub-space #dims	# models r	upper bound $\phi^{(\text{NMI})}$ (κ,κ)	all features $\phi^{(\text{NMI})}$ $(\kappa,\lambda^{(\text{all})})$	**consensus** $\phi^{(\text{NMI})}$ (κ,λ)	max subspace $\max_q \phi^{(\text{NMI})}$ $(\kappa,\lambda^{(q)})$	average subs. $\text{avg}_q \phi^{(\text{NMI})}$ $(\kappa,\lambda^{(q)})$	min subspace $\min_q \phi^{(\text{NMI})}$ $(\kappa,\lambda^{(q)})$
2D2K	1	3	1.00000	0.84747	**0.68864**	0.68864	0.64145	0.54706
8D5K	2	5	1.00000	1.00000	**0.98913**	0.76615	0.69822	0.62134
PENDIG	4	10	0.99736	0.67715	**0.59009**	0.53197	0.44625	0.32598
YAHOO	128	20	0.86602	0.44763	**0.38167**	0.21403	0.17075	0.14582

Table 5.3: Feature-Distributed Clustering (FDC) results. The consensus clustering is as good as or better than the best individual subspace clustering for all four data-sets.

We also conducted FDC experiments on the other three data-sets. Table 5.3 summarizes the results and several comparison benchmarks. The choice of the number of random subspaces r and their dimensionality is currently driven by the user. For example, in the YAHOO case, 20 clusterings were performed in 128-dimensions (occurrence frequencies of 128 random words) each. The average quality amongst the results was 0.17 and the best quality was 0.21. Using the supra-consensus function to combine all 20 labelings yields a quality of 0.38, or 124% higher mutual information than the average individual clustering. In all scenarios, the consensus clustering is as good or better than the best individual input clustering and always better than the average quality of individual clusterings. When processing on the all features is not possible

and only limited views exist, a cluster ensemble can boost results significantly compared to individual clusterings. Also, since the combiner has no feature information, the consensus clustering tends to be poorer than the clustering on all features. However, as discussed in the introduction, in knowledge-reuse application scenarios, the original features are unavailable, so a comparison to an all-feature clustering is only done as a reference.

The supra-consensus function chooses either MCLA and CSPA results but the difference is not statistically significant. As noted before MCLA is much faster and should be the method of choice if only one is needed. HGPA delivers poor ANMI for `2D2K` and `8D5K` but improves with more complex data in `PENDIG` and `YAHOO`. However, MCLA and CSPA performed significantly better than HGPA for all four data-sets.

5.4.5 Object-Distributed Clustering (ODC)

A dual to the application described in the previous section, is Object-Distributed Clustering (ODC). In this scenario, individual clusterers have a limited selection of the object population but have access to all the features of the objects it is provided with.

This is somewhat more difficult than FDC, since the labelings are partial. Because there is no access to the original features, the combiner Γ needs some overlap between labelings to establish a meaningful consensus[11]. Object-distribution can naturally result from operational constraints in many application scenarios. For example, datamarts of individual stores of a retail company may only have records of visitors to that store, but there are enough people

[11] When features are available as in previously considered distributed data mining scenarios [KPHJ99], one can e.g., merge partitions based on their centroids to reach consensus.

who visit more than one store of that company to result in the desired overlap. On the other hand, even if all the data is centralized, one may artificially 'distribute' them in the sense of running clustering algorithms on different but overlapping samples (record-wise partitions) of the data, and then combine the results as this can provide a computational speedup when the individual clusterers have super-linear time complexity.

In this subsection, we will discuss how one can use consensus functions on overlapping sub-samples. We propose a wrapper to any clustering algorithm that simulates a scenario with distributed objects and a combiner that does not have access to the original features. In ODC, we introduce an *object partitioning* and a corresponding *clustering merging* step. The actual clustering is referred to as inner loop clustering. In the pre-clustering partitioning step Π, the entire set of objects $\mathcal{X}$ is decomposed into p overlapping partitions π:

$$\mathcal{X} \xrightarrow{\Pi} \{\pi^{(q)} \mid q \in \{1, \ldots, p\}\} \tag{5.7}$$

Two partitions $\pi^{(a)}$ and $\pi^{(b)}$ are *overlapping* iff $|\pi^{(a)} \cap \pi^{(b)}| > 0$.[12] Also, a set of partitions provides *full coverage* iff $\mathcal{X} = \bigcup_{q=1}^{p} \pi^{(q)}$.

The ODC framework is parametrized by p, the number of partitions, and l, the degree of overlap ranging between 0 and 1. No overlap, or $l = 0$, yields disjoint partitions and, by design, $l = 1$ implies p-fold fully overlapping (identical) samples. Instead of l, one can use the parameter $v > 1$ to fix the total number of points processed in all p partitions combined to be (approximately) vn. This is accomplished by choosing $l = (v - 1)/(p - 1)$.

Let us assume that the data is not ordered, so any contiguous indexed subsample is equivalent to a random subsample. Every partition should have

[12]Overlap assures that the meta-graph in MCLA is connected, e.g. a path between any pair of hyperedges/clusters exists.

the same number of objects for load balancing. Thus, in any partition π there are $|\pi| = \lceil \frac{p-1}{p} nl \rceil + \lceil \frac{n}{p} \rceil$ objects. Now that we have the number of objects $|\pi|$ in each partition, let us propose a simple coordinated sampling strategy: For each partition there are $\lceil n/p \rceil$ objects deterministically picked so that the union of all p partitions provides full coverage of all n objects. The remaining objects for a particular partition are then picked randomly without replacement from the objects not yet in that partition. There are many other ways of coordinated sampling. In this chapter we will limit our discussion to this one strategy for brevity.

Each partition is processed by independent, identical clusterers (chosen appropriately for the application domain). For simplicity, we use the same k in the sub-partitions. The post-clustering merging step is done using our supra-consensus function Γ.

$$\{\lambda^{(q)} \mid q \in \{1, \ldots, p\}\} \xrightarrow{\Gamma} \lambda \tag{5.8}$$

Since every partition only looks at a fraction of the data, there are missing labels in the $\lambda^{(q)}$'s. Given sufficient overlap, the supra-consensus function Γ ties the individual clusters together and delivers a consensus clustering.

We performed the following experiment to demonstrate how the ODC framework can be used to perform clustering on partially overlapping samples without access to the original features. We use graph partitioning as the clusterer in each processor and vary the number of partitions from 2 to 72 with $v = 2$. Figure 5.11 shows our results for the four data-sets. Each plot in figure 5.11 shows the relative mutual information (fraction of mutual information retained as compared to the reference clustering on all objects and features) as a function of the number of partitions. We fix the sum of the number of

objects in all partitions to be double the number of objects ($v = 2$). Within each plot, p ranges from 2 to 72 and each ODC result is marked with a '$\circ$'. The reference point for unpartitioned data is marked by '$\times$' at (1,1). For each of the plots, we fitted a sigmoid function to summarize the behavior of ODC for that scenario.

Clearly, there is a tradeoff in the number of partitions versus quality. As p approaches vn, each clusterer only receives a single point and can make no reasonable grouping. For example, in the `YAHOO` case, for $v = 2$ processing on 16 partitions still retains around 90% (80%) of the full quality. For less complex data-sets, such as `2D2K`, combining 16 partial partitionings ($v = 2$) still yields 90% of the quality. In fact, `2D2K` can be clustered in 72 partitions at 80% quality. Also, we observed that for easier data-sets there is a smaller *absolute* loss in quality for more partitions p.

Regarding our proposed techniques, all three algorithms achieved similar ANMI scores without significant differences for `8D5K`, `PENDIG`, and `YAHOO`. HGPA had some instabilities for the `2D2K` data-set delivering inferior consensus clusterings compared to MCLA and CSPA.

In general, we believe the loss in quality with p has two main causes. First, through the reduction of considered pairwise relations, the problem is simplified as speedup increases. At some point too much relationship information is lost to reconstruct the original clusters. The second factor is related to the balancing constraints used by the graph partitioner in the inner loop: the sampling strategies cannot maintain the balancing, so enforcing them in clustering hurts quality. A relaxed inner loop clusterer might improve results.

Distributed clustering using a cluster ensemble is particularly useful

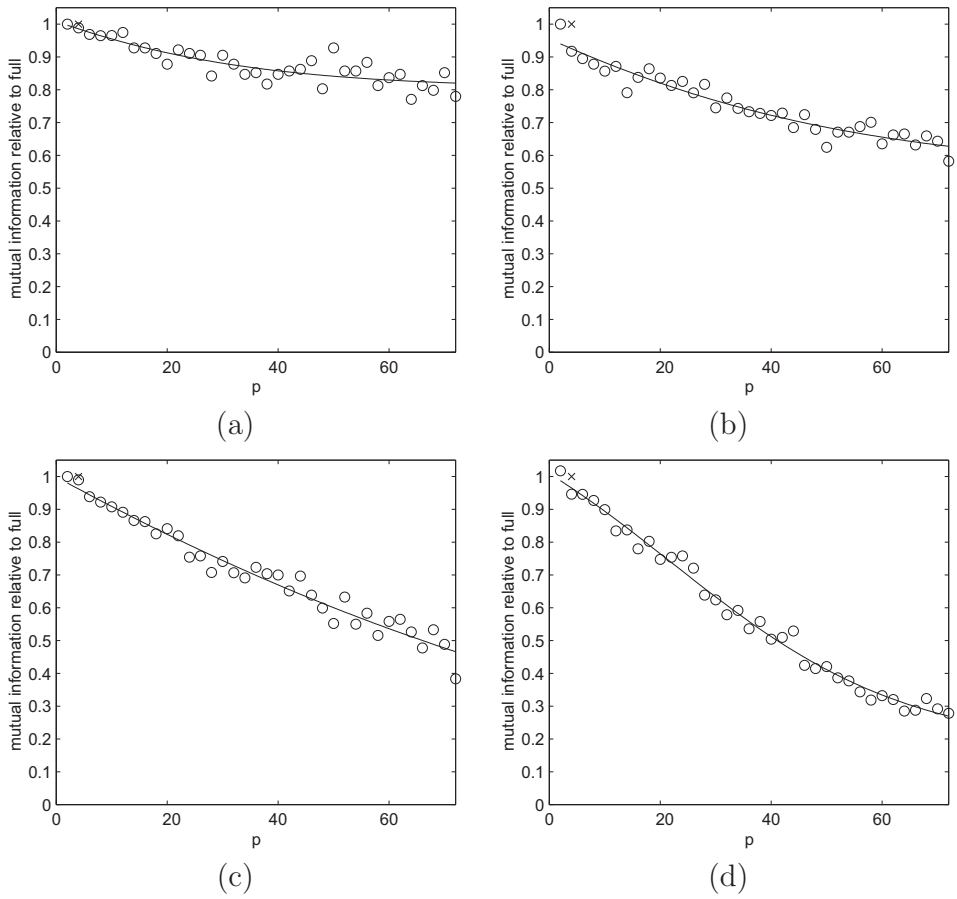


Figure 5.11: Object-Distributed Clustering (ODC) results. Clustering quality (measured by relative mutual information) as a function of the number of partitions, p, on various data-sets: (a) `2D2K`; (b) `8D5K`; (c) `PENDIG`; (d) `YAHOO`. The sum of the number of samples over all partitions is fixed at $2n$. Each plot contains experimental results using graph partitioning in the inner loop for $p = [2, \ldots, 72]$ and a fitted sigmoid (least squared error). Processing time can be reduced by a factor of 4 for the `YAHOO` data while preserving 80% of the quality ($p = 16$).

when the inner loop clustering algorithm has superlinear complexity ($> O(n)$) and a fast consensus function (such as MCLA and HGPA) is used. In this case, additional speedups can be obtained through distribution of objects. Let us assume that the inner loop clusterer has a complexity of $O(n^2)$ (e.g., similarity-based approaches or efficient agglomerative clustering) and one uses only MCLA and HGPA in the supra-consensus function.[13] We define speedup as the computation time for the full clustering divided by the time when using the ODC approach. The overhead for the MCLA and HGPA consensus functions grows linear in n and is negligible compared to the $O(n^2)$ clustering. Hence the sequential speedup is approximately $s^{(\mathrm{ODC-SEQ})} = \frac{p}{v^2}$. Each partition can be clustered without any communication on a separate processor. At integration time only the n-dimensional label vector (instead of e.g., the entire $n \times n$ similarity matrix) has to be transmitted to the combiner. Hence, ODC does not only save computation time, but also enables trivial p-fold parallelization. Consequently the sequential speedup can be multiplied by p if a p-processor computer is utilized: $s^{(\mathrm{ODC-PAR})} = \frac{p^2}{v^2}$. An actual speedup ($s > 1$) will be reached for $p > v^2$ in the sequential or $p > v$ in the parallel case. For example, when $p = 4$ and $v = 2$ (which implies $l = 1/3$) the computing time is approximately the same ($s = 1$) because each partition is half the original size $n/2$ and consequently processed in a quarter of the time. Since there are four partitions, ODC takes the same time as the original processing. In our experiments, using partitions from 2 to 72 yield corresponding sequential (parallel) speedups from 0.5 (1) to 18 (1296). For example, `2D2K` (`YAHOO`) can be sped up 64-fold using 16 processors at 90% (80%) of the full length quality. In fact, `2D2K` can be clustered in less that 1/1000 of the original time at 80% quality.

[13] CSPA is $O(n^2)$ and would reduce speedups obtained by distribution.

5.5 Summary

In this chapter we introduced the cluster ensemble problem and provided three effective and efficient algorithms to solve it. We defined a mutual information based objective function that enables us to automatically select the best solution from several algorithms and allows one to build a supra-consensus function as well. We conducted experiments to show how cluster ensembles can be used to introduce robustness, speed-up superlinear clustering algorithms, and dramatically improve 'sets of subspace clusterings' for a large variety of domains. In document clustering of Yahoo! web-pages we showed that combining e.g. 20 clusterings each obtained from only 128 random words can more than double quality compared to the best single result. Some of the algorithms and datasets are available for download at `http://strehl.com/`. Indeed, the cluster ensemble is a very general framework and enables a wide range of applications. We are especially interested in new applications for knowledge reuse and for distributed clustering.

Chapter 6

Concluding Remarks

I never think of the future. It comes soon enough.

– Albert Einstein[1]

6.1 Summary of Contributions

In this dissertation, cluster analysis was extended in several directions driven by information-rich, complex data. Real-life objects are often characterized by an abundance of features as well as taxonomies from a variety of views and times. The main contributions of this dissertation are summarized in figure 6.1. The contributions are organized along three areas of cluster analysis activities, namely applications, algorithms, and result assessment.

Transactional shopping records motivated us to develop better analysis tools than the available tools such as a-priori association rule mining or k-means. We developed a *relationship-based clustering framework* that relies only on pairwise similarities to side-step the curse of dimensionality. In this spirit,

[1] In interview given aboard the liner *Belgenland*, New York, December 1930

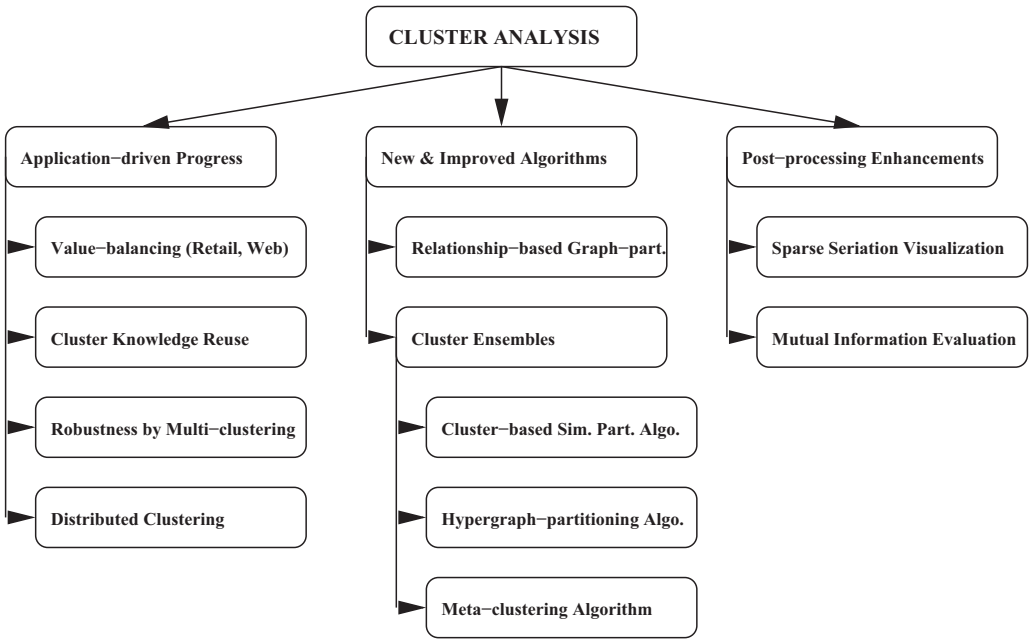


Figure 6.1: Summary of contributions in this dissertation.

we developed an intuitive clustering visualization using *sparse seriation* of the pairwise similarity matrix. We improved similarity measures by proposing and analyzing an extended Jaccard similarity measure. In the retail domain, we also introduced application-driven constraints of *value-balancing* to cluster analysis (e.g., clusters have approximately equal total revenue or number of customers). These constraints turned out to be useful in a variety of other domains such as clustering web-sessions and text documents.

In our work with text document data, the availability of human engineered document taxonomies inspired a *clustering performance evaluation measure based on mutual information*. The results obtained through this framework showed that relationship-based clustering is indeed superior when

appropriate domain-specific similarities are chosen.

When a multitude of taxonomies is already present, one might just want to *reuse such existing knowledge* and integrate previous results instead of starting from scratch. This led us to develop *cluster ensembles*, an approach to adopt multi-learner systems for clustering. We proposed a formal cluster ensemble problem and developed three effective and efficient algorithms to solve it. This combiner framework is useful in a variety of applications besides knowledge reuse. For example, it enables *distributed clustering* and can provide *robustness through multiple clusterings*.

6.2 Future Work

There are several directions for future research on relationship-based learning frameworks. This section highlights some of the promising directions. We will first discuss some theoretical and algorithmic improvements of cluster ensembles and conclude with a selection of particularly interesting application domains.

6.2.1 Greedy Maximization for Cluster Ensembles

The Average Normalized Mutual Information (ANMI) objective defines the purpose of the cluster ensemble as proposed in chapter 5. We desire a more thorough theoretical analysis of ANMI to better understand its properties and develop better optimization schemes. Investigating the ANMI objective might also result in a better understanding of the biases of the three proposed consensus functions.

Based on this, we plan to explore greedy optimization schemes. As an enhancement for our proposed consensus functions in applications where n is not too large, a greedy post-processing step can be used to refine the best labeling of the three algorithms from section 5.2 as follows: For each object, we exchange the current label with any of the $k-1$ possible other labels and evaluate the ANMI objective. If the ANMI increases, we change the object's label to the best new value and repeat the trials. Thereby we greedily optimize the objective through single label changes. When for all objects there is no label change that improves the objective, a local optimum has been reached and this is the final consensus labeling. Like all local optimization procedures, there is a strong dependency on the initialization. Running this greedy search starting with a random labeling is often computationally intractable and tends to result in poor local optima. However, using the best of the three consensus functions as an initialization, the greedy search might work well. Preliminary experiments indicate that this post-processing affects between 0% - 5% of the labels and yields slightly improved results. Adaptive step sizes (e.g., modifying larger label segments / blocks) and less local search techniques (e.g., maintaining a population of current consensus clusterings instead of just one) might be promising to investigate for greedy search.

Related work on integer programming techniques, Independent Component Analysis (ICA), and Maximum Mutual Information (MMI) techniques might also yield insights towards direct optimization of the objective.

We also explored extending the ANMI objective to be computed from sets of soft clusterings which yields a smooth (numerically differentiable) input space. However, preliminary experiments using gradient ascent techniques on the continuous cluster association input space failed. This is probably due to

the objective not being well behaved and the large dimensionality of the input space for optimization.

6.2.2 Soft Cluster Ensembles

While the three consensus functions proposed are designed to also work with soft clustering inputs, we have not conducted an experimental study to evaluate their behavior in that scenario. Possibly, soft clustering ensembles will perform superior to hard clustering ensembles since the confidences on cluster memberships can be leveraged to combine clusterings in a risk-aware fashion.

One can also extend the combiners to output soft consensus clusterings. In fact, the Meta-CLustering Algorithm (MCLA) already computes soft consensus clusterings, and the Cluster-based Similarity Partitioning Algorithm (CSPA) and the HyperGraph Partitioning Algorithm (HGPA) can be extended to do so.

6.2.3 Feature-augmented Cluster Ensembles

In cluster ensembles, we reuse only previous labelings and do not allow access to the original features. In many application, at least limited information from the original features is available at the combination stage. Primarily this is the case when the set of clusterings is generated by intentional splitting of the data.

Using extra feature information might further boost results and allow the cluster ensemble to be applied in a wider class of knowledge reuse applications. For example, in distributed clustering the minimum information for consolidation are the individual labelings. Adding extra feature information

such as cluster centroids to augment the cluster ensemble can possibly improve results significantly.

6.2.4 Distributed Clustering

In this dissertation, we propose cluster ensembles and show how this allows distributed clustering *without* sharing of actual features. The ability to do this analysis without access to the primary data opens opportunities for performing data mining on knowledge maintained in multiple clients without them revealing their data to other clients. Despite these restrictions, the combiner can discover knowledge that could not be discovered by any single client. Cluster ensembles can be used to develop a federated data mining system that ensures privacy and security without requiring that clients send their data to the system or to other clients.

For federated systems, we want to extend our application scenarios. In real scenarios, a variety of hybrids of the investigated RCC, FDC, and ODC scenarios can be encountered. Cluster ensembles could enable federated data mining systems working on top of distributed and heterogeneous databases.

In particular, one can study the impact of the coordinated subsampling strategies on the performance and quality of object distributed clustering. The question is to determine what types of overlap and object ownership structures lend themselves particularly well for knowledge reuse.

A key requirement of data mining techniques is that they must scale to large-scale data-sets. Distributed processing can be used as a tool to scale complex algorithms to large data-sets. As discussed in chapter 5, cluster ensembles can be used in such a scenario to trade-off quality for speed. The

proposed cluster ensemble requires no communication during the individual grouping. However, when extending cluster ensembles such that individual clusterers share object features in the inner loop as well as with the combining stage, several designs of inter-processor communication can be made. This might make it interesting to revisit the speedup versus quality tradeoff.

6.2.5 Bioinformatics

Molecular biologists are currently engaged in some of the most impressive data collection projects. Recent genome-sequencing projects are generating an enormous amount of data related to the function and structure of biological molecules and sequences. The interpretation of this wealth of data may deeply affect our understanding of life at the molecular level. Important problems for which cluster analysis might be very successful include the prediction of protein structure and function, semi-automated drug design, interpretation of nucleotide sequences, and knowledge acquisition from genetic data.

In micro-array data, for example, an object could be a gene, while a feature could be the level of expression of that gene under a particular condition. There typically are thousands of genes under hundreds of conditions. This data shares many of the properties of the high-dimensional data investigated in this dissertation and the relationship-based clustering approach proposed in chapters 3 and 4 seems promising for finding interesting genes, for example.

Also, the multitude of experimental results available to industrial gene expression researchers warrants the investigation of applications of cluster ensembles as proposed in chapter 5 to integrate and consolidate previous partitions of genes by function. Thereby cluster ensembles can yield robust results

that ‘smooth’ out variations in the individual experiments without requiring researchers to integrate their entire primary data.

Appendix A

Data-sets

Data is a lot like humans: It is born. Matures. Gets married
to other data, divorced. Gets old. One thing that it
doesn't do is die. It has to be killed.
– Arthur Miller[1]

In the following, some of the data-sets used in this dissertation are illustrated.

[1]In speech on Annual Seminar of the *American Society for Industrial Security*, Boston, September 1988

A.1 Gaussian Data-sets

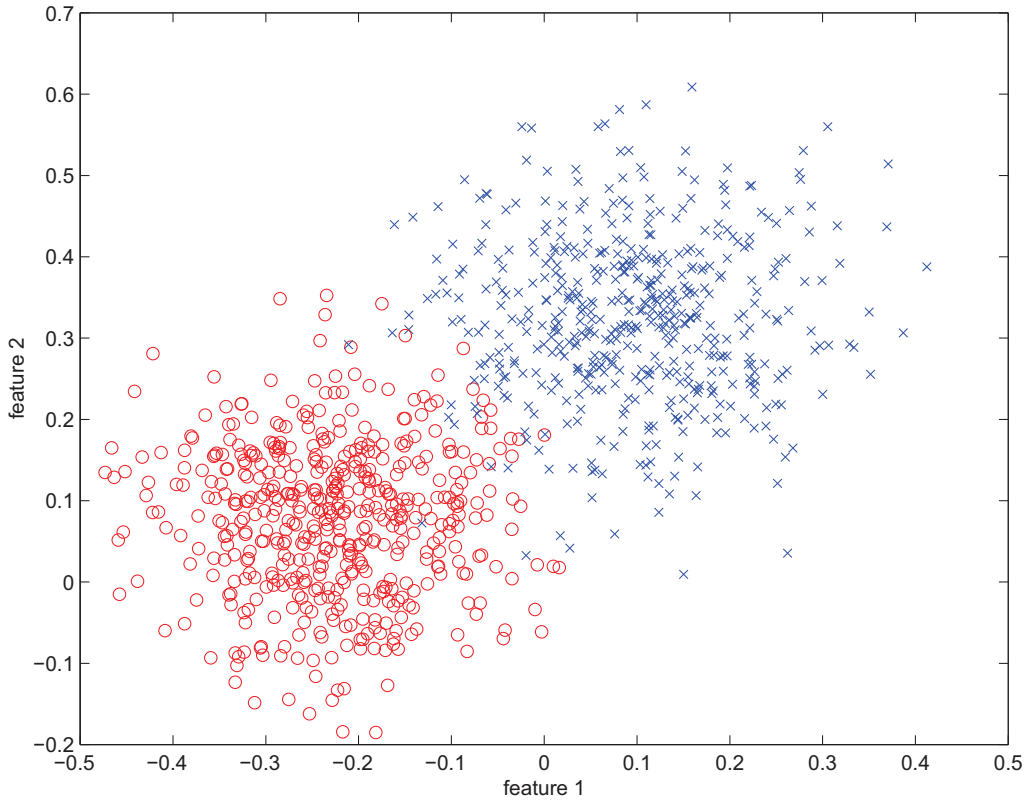


Figure A.1: `2D2K` data-set. The two Gaussians overlap.

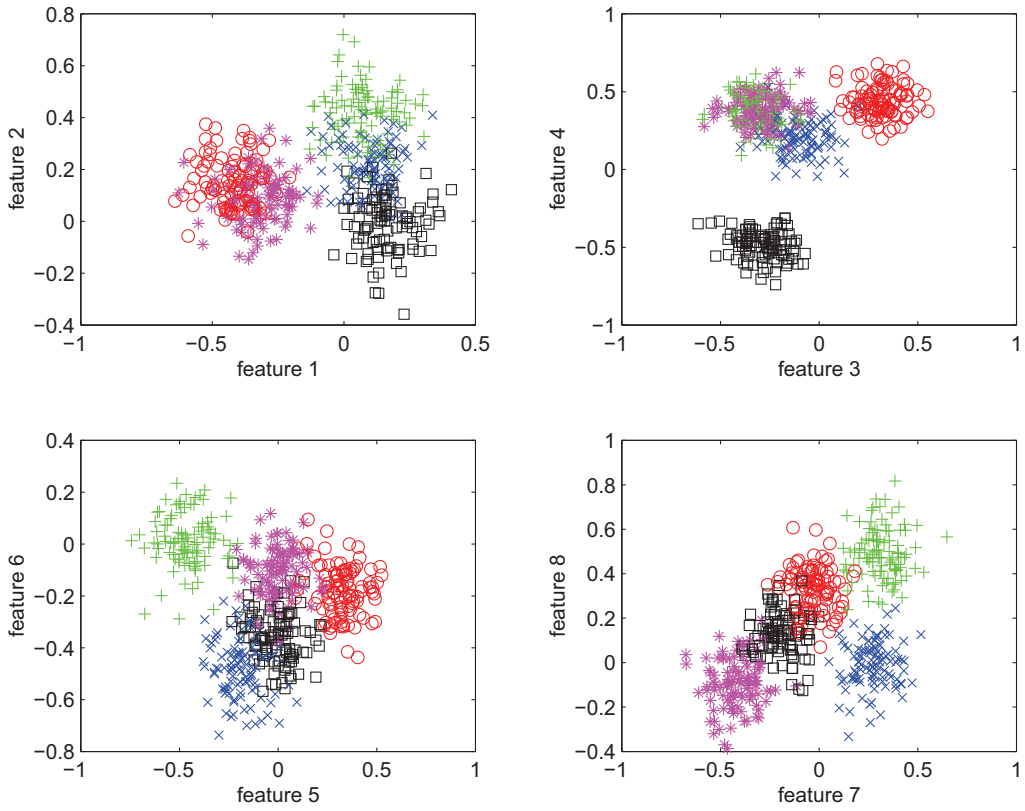


Figure A.2: `8D5K` data-set. While the Gaussians overlap in the partial 2D views shown, they are not overlapping in the full 8D space.

A.2 Iris Data-set

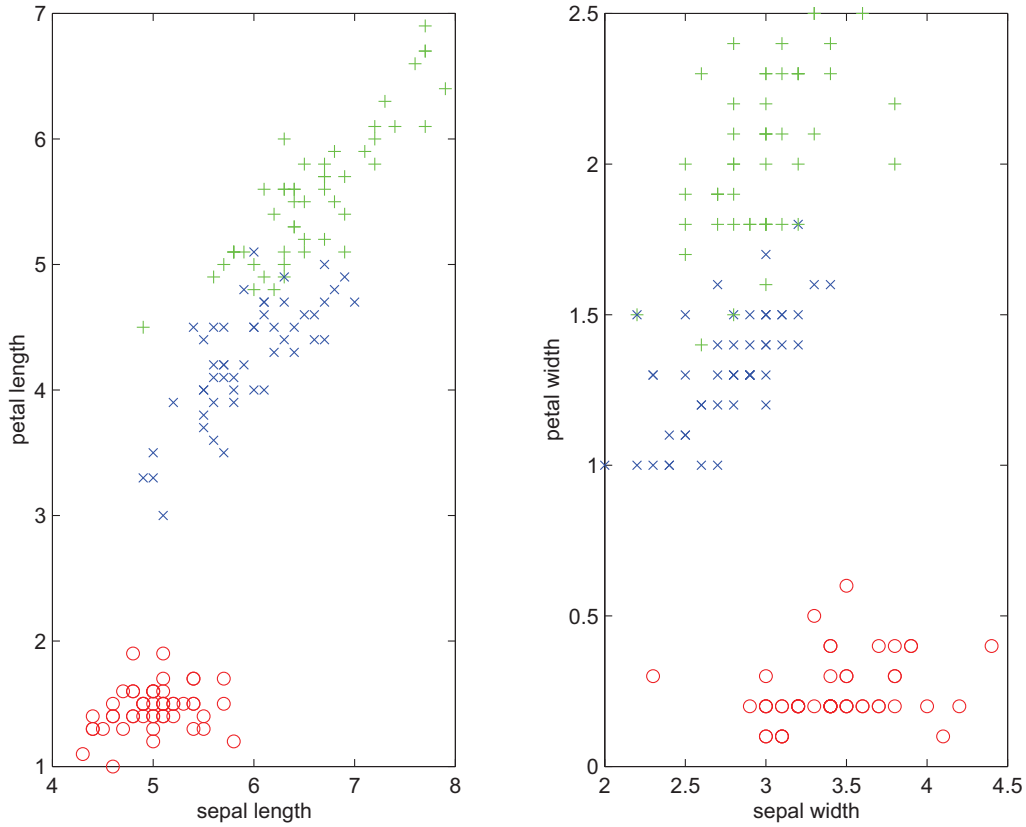


Figure A.3: `IRIS` data-set. The data contains 4 measurements taken for each of 150 iris flowers.

A.3 Pen Digits Data-set

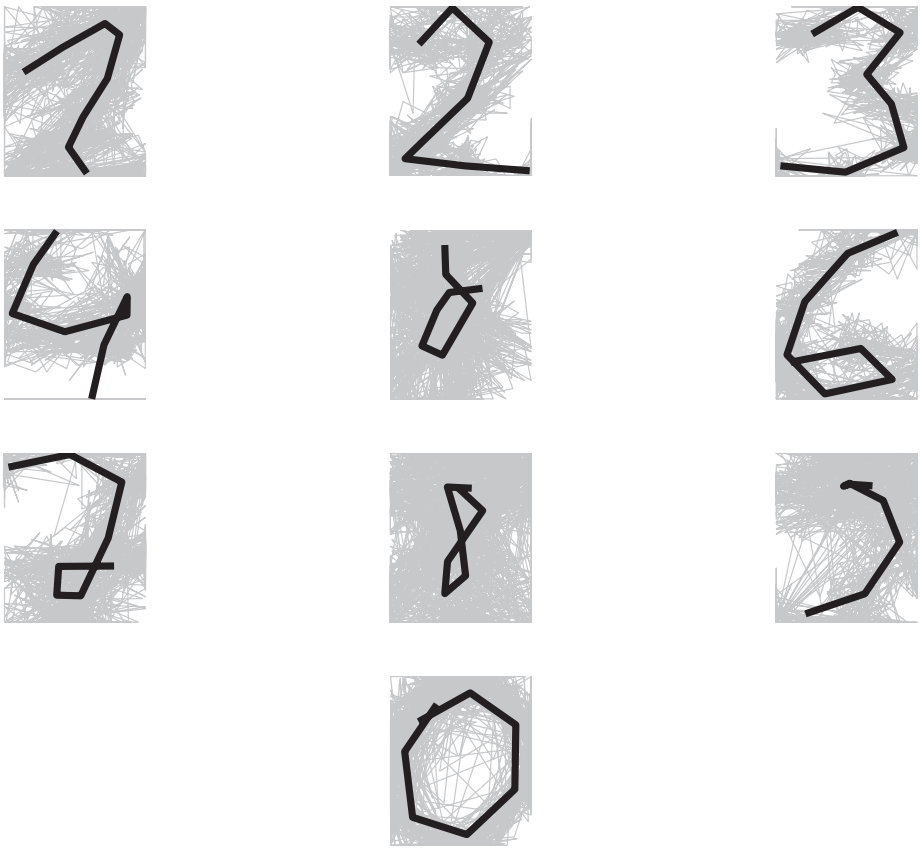


Figure A.4: Pen digits data-set `PENDIG`. 1500 (of 7494) samples of the handwritten numerals 0-9 are plotted in gray arranged by their classes. Their mean is shown superimposed with bold black.

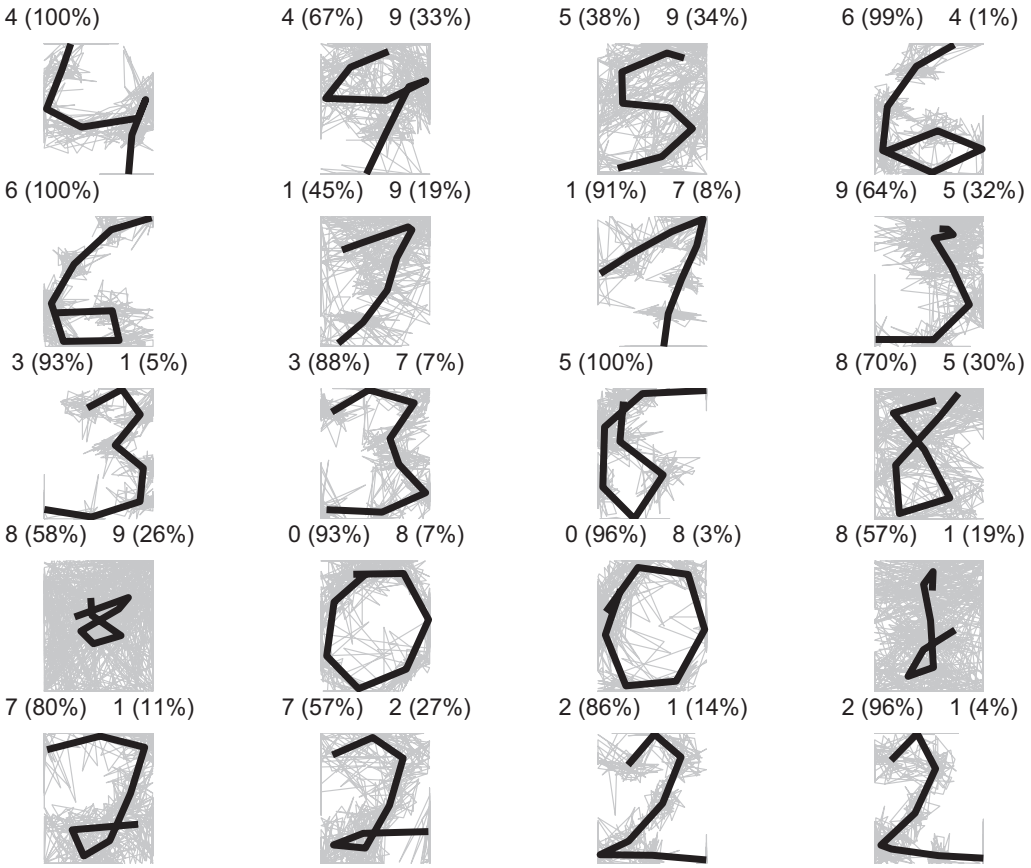


Figure A.5: `PENDIG` clusters. 1500 (of 7494) samples of the handwritten numerals 0-9 are illustrated arranged by clusters. Strong intra-class variation as e.g., seen in the numeral 5 is disambiguated into several clusters: 5 can be written like a 'S' in one stroke from top to bottom or in two strokes with the top dash last.

A.4 Drugstore Data-set

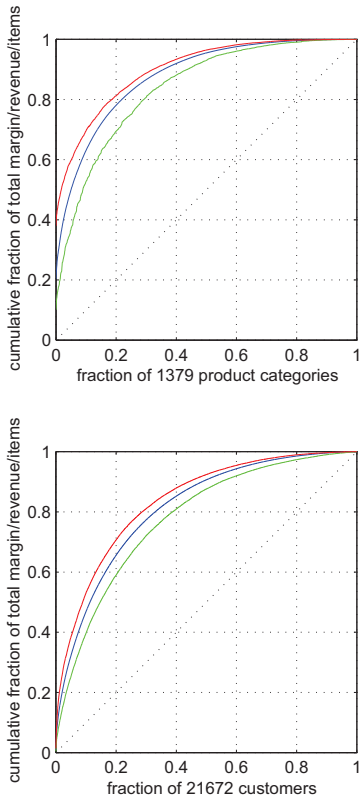


Figure A.6: Drugstore data before preprocessing. Distribution of profit margin (red, highest), revenue (blue, between), item count (green, lowest), and profit margin (red) for 21672 active customers and 1379 purchased product categories. Sorted by revenue per product (top) and per customer (bottom). Total revenue on 543,744 items sold is $1,974,824.52 and total profit margin is $908,302.88.

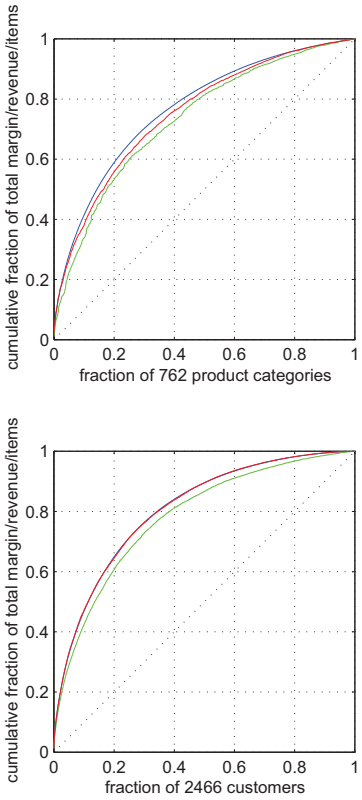
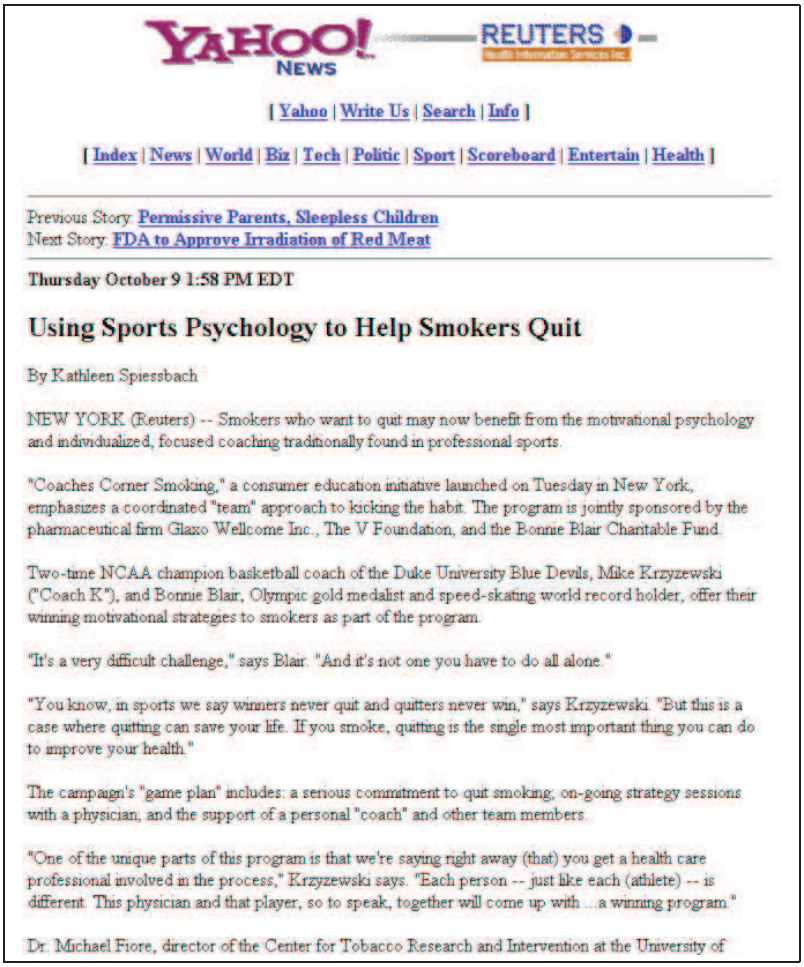


Figure A.7: Drugstore data sample `RETAIL`. Margin/revenue/item count distribution for 762 selected product categories (top) and 2466 sampled customers (bottom). Total revenue on 38,816 items sold is \$126,899.03 and total profit is \$45,151.18.

```
Customer with rank 333/2466
 1 @  4.99 CHRISTMAS GIFTWARE ITEMS 621
 1 @  2.95 AMERICAN GREETINGS CARDS ITEMS 1317
 1 @  1.49 CHRISTMAS CARDS ITEMS 74
 3 @  2.49 BASKET CANDY ITEMS 129
 1 @  6.19 MAYBELLINE EYE ITEMS 234
 3 @  1.99 CHRISTMAS WRAPPING PAPER ITEMS 118
 1 @  5.19 TAMPONS ITEMS 153
 1 @  5.99 CONDITIONER MIDDLE PRICE ITEMS 215
 2 @  1.69 VALENTINE BOX CANDY ITEMS 65
 3 @  0.99 CHRISTMAS GARLAND, ETC ITEMS 55
 2 @  0.29 CHRISTMAS BOWS, RIBBONS ITEMS 75
48 @  0.19 CHRISTMAS CANDLES ITEMS 134
 1 @  9.99 APPLIANCES HAIR STYLERS ITEMS 56
 1 @  6.59 MAYBELLINE FACE ITEMS 228
 2 @  2.99 CHRISTMAS ORNAMENTS ITEMS 153
 1 @  0.64 SUMMER TOYS ITEMS 355
 1 @  0.89 GERMICIDAL ANTISEPTICS ITEMS 24
 2 @  2.24 VALENTINE BOX CARDS ITEMS 89
 1 @  4.59 FEM HY FEM PAIN ITEMS 21
 1 @  4.99 EXTENSION CORDS ITEMS 19
 2 @  1.72 NN KNEE HIGHS ITEMS 25

Customer with rank 867/2466
 2 @ 11.04 FRAGRANCES   OPEN STOCK ITEMS 361
 2 @  4.99 SHOWER GEL & BODY WASH ITEMS 313
 1 @  3.19 TOILET TISSUE ITEMS 50
 4 @  0.99 HALLOWEEN CANDLES/IMPORT ITEMS 78

Customer with rank 1419/2466
 1 @  8.19 VITAMINS MULTI ITEMS 155
 1 @  7.99 VITAMINS C ITEMS 129
 1 @  1.89 SCHOOL ACCESSORIES ITEMS 263

Customer with rank 2089/2466
 1 @  1.59 TAPE ITEMS 122
 1 @  4.99 TP PICTURE FRAMES/HANGERS ITEMS 1423
```

Table A.1: Exemplary market-baskets from drugstore data-set `RETAIL`.

A.5 Yahoo! News Data-set



YAHOO! NEWS — REUTERS Health Information Services Inc.

[Yahoo | Write Us | Search | Info]

[Index | News | World | Biz | Tech | Politic | Sport | Scoreboard | Entertain | Health]

Previous Story: Permissive Parents, Sleepless Children
Next Story: FDA to Approve Irradiation of Red Meat

Thursday October 9 1:58 PM EDT

Using Sports Psychology to Help Smokers Quit

By Kathleen Spiessbach

NEW YORK (Reuters) -- Smokers who want to quit may now benefit from the motivational psychology and individualized, focused coaching traditionally found in professional sports.

"Coaches Corner Smoking," a consumer education initiative launched on Tuesday in New York, emphasizes a coordinated "team" approach to kicking the habit. The program is jointly sponsored by the pharmaceutical firm Glaxo Wellcome Inc., The V Foundation, and the Bonnie Blair Charitable Fund.

Two-time NCAA champion basketball coach of the Duke University Blue Devils, Mike Krzyzewski ("Coach K"), and Bonnie Blair, Olympic gold medalist and speed-skating world record holder, offer their winning motivational strategies to smokers as part of the program.

"It's a very difficult challenge," says Blair. "And it's not one you have to do all alone."

"You know, in sports we say winners never quit and quitters never win," says Krzyzewski. "But this is a case where quitting can save your life. If you smoke, quitting is the single most important thing you can do to improve your health."

The campaign's "game plan" includes: a serious commitment to quit smoking; on-going strategy sessions with a physician; and the support of a personal "coach" and other team members.

"One of the unique parts of this program is that we're saying right away (that) you get a health care professional involved in the process," Krzyzewski says. "Each person -- just like each (athlete) -- is different. This physician and that player, so to speak, together will come up with ...a winning program."

Dr. Michael Fiore, director of the Center for Tobacco Research and Intervention at the University of

Figure A.8: Exemplary news web-page from YAHOO data-set.

A.6 20 Newsgroup Data-set

```
From: pjsinc@phoenix.oulu.fi (Petri Salonen)
Subject: Re: What does the .bmp format mean?

Michael Panayiotakis (louray@seas.gwu.edu) wrote:
: In article <robertsa@unix2.tcd.ie> (Andrew L. Roberts) writes:
: >What exactly does the windows bitmap format look like? I mean, how is
: >the data stored: width, height, no. of colours, bitmap data? I couldn't
: >find anything in ths user manual, is there any other reference material
: >which would give me this information?

: Well, this is *only* a guess:  If it goes by the "true" meaning of "bit
: map", then it holds (x,y,c) where x pixel number in th ex-direction, y:
: pixel-number in the y-dir, c: colour.

Come on fellows! The format is quite plainly explained in the manuals.
It's in the "Programmer's Reference, Volume 3: Messages, Structures,
and Macros" (MSC-Dev.kit for 3.1, should be also in the Borland's
manuals) pages 232-241 (depending what you need).

First there is the BITMAPFILEHEADER-struct then the BITMAPINFO which
contains the BITMAPINFOHEADER and the RGBQUAD and then the bitmap
data. AND there is also a example among the example files (MS_SDK).
Hope this helps....

--------------------------------------------------------------------------
    ########################## | Yes, I do have some prior knowledge in this.
   ########################## | There is nothing dangerous in these dragons,
  ####   / ///   /            | they are totally harmless... But my opinion
 ####   /  /    / /// ///     | is that kicking them might not be the right
#### /// ///   / / / /// /    | way to test it. So shut up and RUN!
--------------------------------------------------------------------------
pjsinc@sunrise.oulu.fi  pjsinc@phoenix.oulu.fi  pjsinc@tolsun.oulu.fi
If it's possible that there are some opinions above, they must be all MINE.
```

Figure A.9: Exemplary newsgroup posting from N20 data-set.

Appendix B

Derivations

Beware of bugs in the above code; I have
only proved it correct, not tried it.
– Donald Knuth[1]

B.1 Normalized Symmetric Mutual Information

Let X and Y be the random variables described by the cluster labeling λ and category labeling κ, respectively. Let $H(X)$ denote the entropy of a random variable X. Mutual information between two random variables is defined as

$$I(X,Y) = H(X) + H(Y) - H(X,Y). \tag{B.1}$$

Also,

$$I(X,Y) = H(X) - H(X|Y) \leq H(X) \tag{B.2}$$

[1]In letter sent to Peter van Emde Boas, March 1977

and

$$I(X,Y) = H(X) - H(Y|X) \leq H(Y). \tag{B.3}$$

So

$$I(X,Y) \leq \min(H(X), H(Y)). \tag{B.4}$$

Since $\min(H(X), H(Y)) \leq \frac{H(X)+H(Y)}{2}$, a tight upper bound on $I(X,Y)$ is given by $\frac{H(X)+H(Y)}{2}$. Thus, a worst case upper bound for all possible labelings A (with labels from 1 to k) and categorizations (with labels from 1 to g) is given by

$$I(X,Y) \leq \max_{A\in\{1,\ldots,k\},B\in\{1,\ldots,g\}} \left(\frac{H(A)+H(B)}{2}\right). \tag{B.5}$$

Hence, we define [0,1]-normalized mutual information-based quality as

$$NI(X,Y) = \frac{2\cdot I(X,Y)}{\max_{A\in\{1,\ldots,k\}}(H(A)) + \max_{B\in\{1,\ldots,g\}}(H(B))}. \tag{B.6}$$

Using

$$I(X,Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x,y)}{p(x)\cdot p(y)}, \tag{B.7}$$

and approximating probabilities with frequency counts delivers our quality measure $\phi^{(\mathrm{NMI})}$:

$$\phi^{(\mathrm{NMI})}(\lambda,\kappa) = \frac{2\cdot\sum_{\ell=1}^{k}\sum_{h=1}^{g}\frac{n_l^{(h)}}{n}\log\frac{n_l^{(h)}/n}{(n^{(h)}/n)\cdot(n_l/n)}}{\log(k)+\log(g)} \tag{B.8}$$

Basic simplifications yield:

$$\phi^{(\mathrm{NMI})}(\lambda,\kappa) = \frac{2}{n}\sum_{\ell=1}^{k}\sum_{h=1}^{g} n_\ell^{(h)}\log_{k\cdot g}\left(\frac{n_\ell^{(h)} n}{n^{(h)} n_\ell}\right) \tag{B.9}$$

This derivation is used to obtain equations 4.25 and 5.2.

Instead of using the worst case upper bound for all possible labelings and categorizations, one can assume that the categorization priors are given.

This allows a less aggressive denominator for normalization: One can use the actual entropies $H(X)$ and $H(Y)$ from the labelings in equation B.4. However, this results in a bias towards high k.

Another variant of normalization uses the actual entropies $H(X)$ and $H(Y)$ instead of $\max_{A\in\{1,\ldots,k\}}(H(A))$ and $\max_{B\in\{1,\ldots,g\}}(H(B))$ in equation B.5. For the results presented in this dissertation, we will use the worst case normalization (equation B.9).

B.2 Normalized Asymmetric Mutual Information

The entropy of either, clustering X and categorization Y, provides another tight bound on mutual information $I(X,Y)$ that can be used for normalization. Since the categorization Y is a stable, user given distribution, let's consider

$$I(X,Y) \leq H(Y). \tag{B.10}$$

Hence, one can alternatively define [0,1]-normalized asymmetric mutual information based quality as

$$NI(X,Y) = \frac{I(X,Y)}{H(Y)} \tag{B.11}$$

which translates into frequency counts as

$$\phi^{(\mathrm{NAMI})}(\lambda,\kappa) = \frac{\sum_{\ell=1}^{k}\sum_{h=1}^{g} n_\ell^{(h)} \log \frac{n_\ell^{(h)} n}{n^{(h)} n_\ell}}{-\sum_{h=1}^{g} n^{(h)} \log \frac{n^{(h)}}{n}}. \tag{B.12}$$

Note that this definition of mutual information is asymmetric and does not penalize overrefined clusterings λ. Also, $\phi^{(\mathrm{NAMI})}$ is biased towards high k.

Bibliography

One can measure the importance of a scientific work by the number of earlier publications rendered superfluous by it.

– David Hilbert[1]

[ABKS99] Michael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: ordering points to identify the clustering structure. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Philadephia, Pennsylvania, USA*, pages 49–60, 1999.

[AK95] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *Integration: The VLSI Journal*, 19:1–18, 1995.

[Bar81] J. A. Barnett. Computational methods for a mathematical theory of evidence. In *Proc. of IJCAI*, pages 868–875, 1981.

[BDSY99] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[1] Quoted in H. Eves, *Mathematical Circles Revisited*, 1971

[BEHW87] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information processing Letters*, 24:377–380, 1987.

[BFR98] P. S. Bradley, U. M. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Knowledge Discovery and Data Mining*, pages 9–15, 1998.

[BG98] Kurt D. Bollacker and Joydeep Ghosh. A supra-classifier architecture for scalable knowledge reuse. In *Proc. Int'l Conf. on Machine Learning (ICML-98)*, pages 64–72, July 1998.

[BG99] Kurt D. Bollacker and Joydeep Ghosh. Effective supra-classifiers for knowledge base construction. *Pattern Recognition Letters*, 20(11-13):1347–52, November 1999.

[BG01] A. Banerjee and J. Ghosh. Clickstream clustering using weighted longest common subsequences. In *Workshop on Web Mining : 1st SIAM Conference on Data Mining*, pages 33–40, April 2001.

[BG02] A. Banerjee and J. Ghosh. On scaling up balanced clustering algorithms. In *Proc. 2nd SIAM Intl. Conf. on Data Mining*, Apr 2002. In press.

[BGG+99] D. Boley, M. Gini, R. Gross, E. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27:329–341, 1999.

[BHR96] Michael W. Berry, Bruce Hendrickson, and Padma Raghavan. Sparse matrix reordering schemes for browsing hypertext. In *Lectures in Applied Mathematics (LAM)*, volume 32, pages 99–123. American Mathematical Society, 1996.

[Bis95] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[BL97] M. J. A. Berry and G. Linoff. *Data Mining Techniques for Marketing, Sales and Customer Support*. Wiley, 1997.

[BLM86] J. P. Barthelemy, B. Laclerc, and B. Monjardet. On the use of ordered sets in problems of comparison and consensus of classifications. *Journal of Classification*, 3:225–256, 1986.

[BM98] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98)*, 1998.

[Bre94] L. Breiman. Bagging predictors. Technical report, TR No. 421, University of California, Berkeley, 1994.

[BY99] Ricardo Baeza-Yates. *Modern Information Retrieval*. Addison Wesley, New York, 1999.

[Car95] Rich Caruana. Learning many related tasks at the same time with backpropagation. In *Advances in Neural Information Processing Systems 7*, pages 657–664, 1995.

[CDF+98] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *AAAI98*, pages 509–516, 1998.

[CG88] G. A. Carpenter and S. Grossberg. The ART of adaptive pattern recognition by a self-organizing neural network. *IEEE Computer*, 21(3):77–90, 1988.

[CG96] S. V. Chakaravathy and J. Ghosh. Scale based clustering using a radial basis function network. *IEEE Transactions on Neural Networks*, 2(5):1250–61, Sept 1996.

[CG01] K. Chang and J. Ghosh. A unified model for probabilistic principal surfaces. *IEEE Trans. PAMI*, 23(1):22–41, Jan 2001.

[Che99] Chaomei Chen. Visualising semantic spaces and author co-citation networks in digital libraries. *Information Processing and Management*, 35:401–420, 1999.

[CKPT92] Douglass R. Cutting, David Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[CS95] P. Chan and S. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Twelfth International Conference on Machine Learning*, pages 90–98, 1995.

[CS96] Peter Cheeseman and John Stutz. Bayesian classification (AutoClass): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI/MIT Press, 1996.

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

[Das94] B. Dasarathy. *Decision Fusion*. IEEE CS Press, Los Alamitos, CA, 1994.

[DCJ+94] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6(6):1289–1301, 1994.

[DDL+90] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Ed.)*. Wiley, New York, 2001.

[Die01] T. G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems*, pages 1–15. LNCS Vol. 1857, Springer, 2001.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

[DM01] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, January 2001.

[DMS98] Inderjit S. Dhillon, Dharmendra S. Modha, and W. Scott Spangler. Visualizing class structure of multidimensional data. In S. Weisberg, editor, *Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics, Minneapolis, MN, May 13–16 1998*, 1998.

[EH81] B. S. Everitt and D. J. Hand. *Finite Mixture Distributions*. Chapman and Hall, London, 1981.

[EKSX96] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on KDD*, pages 226–231, 1996.

[ESBB98] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, 95:14863–14868, December 1998.

[Eve74] Brian Everitt. *Cluster Analysis*. Heinemann Educational Books, London, 1974.

[FBY92] W. B. Frakes and R. Baeza-Yates. *Information Retrieval : Data Structures and Algorithms*. Prentice-Hall, New Jersey, 1992.

[Fie73] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

[Fie75] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25:619–633, 1975.

[Fis87a] Douglas Fisher. Cobweb: Knowledge acquisition via conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[Fis87b] Douglas Fisher. Improving inference through conceptual clustering. In *National Conference on Artificial Intelligence*, pages 461–465, 1987.

[FL95] C. Faloutsos and K. Lin. Fastmap: a fast algorithm for indexing, data mining and visualization of traditional and multimedia datasets. In *Proc. ACM SIGMOD Int. Conf. on Management of Data, San Jose, CA*, pages 163–174. ACM, 1995.

[FM82] C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. In *Proceedings of the 19th IEEE/ACM Design Automation Conference*, pages 175–181, 1982.

[Fra92] W. Frakes. Stemming algorithms. In W. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 131–160. Prentice Hall, New Jersey, 1992.

[FRB98] U. M. Fayyad, C. Reina, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 194–198. AAAI Press, August 1998.

[Fri94] J. H. Friedman. An overview of predictive learning and function approximation. In V. Cherkassky, J.H. Friedman, and H. Wechsler, editors, *From Statistics to Neural Networks, Proc. NATO/ASI Workshop*, pages 1–61. Springer Verlag, 1994.

[Fuk72] K. Fukanaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1972.

[GBC92] J. Ghosh, S. Beck, and C. C. Chu. Evidence combination techniques for robust classification of short-duration oceanic signals. In *SPIE Conf. on Adaptive and Learning Systems, SPIE Proc. Vol. 1706*, pages 266–276, Orlando, Fl., April 1992.

[GC85] M. A. Gluck and J. E. Corter. Information, uncertainty, and the utility of categories. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pages 283–287, Irvine, CA, 1985. Lawrence Erlbaum Associates.

[GCSR95] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall, London, 1995.

[GG01] G. K. Gupta and J. Ghosh. Detecting seasonal trends and cluster motion visualization for very high dimensional transactional data. In *Proc. First Siam Conf. On Data Mining, (SDM2001)*, pages 115–129, 2001.

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

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, San Francisco, CA, 1979.

[GMT95] J. R. Gilbert, G. L. Miller, and S. Teng. Geometric mesh partitioning: Implementation and experiments. In *Proceedings of the 9th International Parallel Processing Symposium*, pages 418–427. IEEE, April 1995.

[Gra89] C. W. J. Granger. Combining forecasts–twenty years later. *Journal of Forecasting*, 8(3):167–173, 1989.

[GRS98] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 73–84, New York, 1998.

[GRS99] S. Guha, R. Rastogi, and K. Shim. Rock: a robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*, 1999.

[GS02] Joydeep Ghosh and Alexander Strehl. Clustering and visualization of retail market baskets. In N. R. Pal and L. Jain, editors, *Knowledge Discovery in Advanced Information Systems*, AIP. Springer, 2002. In press.

[GSG99] Gunjan K. Gupta, Alexander Strehl, and Joydeep Ghosh. Distance based clustering of association rules. In *Proc. ANNIE 1999, St. Louis*, volume 9, pages 759–764. ASME, November 1999.

[Har75] John A. Hartigan. *Clustering Algorithms*. Wiley, New York, 1975.

[Has93] S. Hashem. *Optimal Linear Combinations of Neural Networks*. PhD thesis, Purdue University, December 1993.

[Hay99] S. Haykin. *Neural Networks: A Comprehensive Foundation*. 2nd edition, Prentice-Hall, New Jersey, 1999.

[HJ97] P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Math. Programming*, 79:191–215, 1997.

[HK00] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[HKKM97] E. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering in a highdimensional space using hypergraph models. Technical Report 97-019, University of Minnesota, Department of Computer Science, 1997.

[HKLK97] Timo Honkela, Samuel Kaski, Krista Lagus, and Teuvo Kohonen. WEBSOM—self-organizing maps of document collections.

In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6*, pages 310–315. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.

[HKT01] J. Han, M. Kamber, and A. K. H. Tung. Spatial clustering methods in data mining: A survey. *Geographic Data Mining and Knowledge Discovery*, 2001.

[HL94] B. Hendrickson and R. Leland. The Chaco user's guide — version 2.0. Technical Report SAND94-2692, Sandia National Laboratories, 1994.

[HL95] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2):452–469, 1995.

[Hub81] P. J. Huber. *Robust Statistics*. Wiley, New York, 1981.

[Ind99] Piotr Indyk. A sublinear-time approximation scheme for clustering in metric spaces. In *Proceedings of the 40th Symposium on Foundations of Computer Science*, 1999.

[JD88] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.

[JH99] T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In M. S. Kearns, S. A. Solla, and D. D. Cohn, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 11, pages 487–493. MIT Press, 1999.

[JK99] E. Johnson and H. Kargupta. Collective, hierarchical clustering from distributed, heterogeneous data. In M. Zaki and C. Ho, editors, *Large-Scale Parallel KDD Systems*, volume 1759 of *Lecture Notes in Computer Science*, pages 221–244. Springer-Verlag, 1999.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[Joa98] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98, Tenth European Conference on Machine Learning*, pages 137–142, 1998.

[KAKS97] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Applications in VLSI domain. In *Proceedings of the Design and Automation Conference*, 1997.

[KC00] H. Kargupta and P. Chan, editors. *Advances in Distributed and Parallel Knowledge Discovery*. AAAI/MIT Press, Cambridge, MA, 2000.

[KG99] S. Kumar and J. Ghosh. GAMLS: A generalized framework for associative modular learning systems. In *Proceedings of the Applications and Science of Computational Intelligence II*, pages 24–34, Orlando, Florida, 1999.

[KHK99] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon:

Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75, August 1999.

[KHSJ01] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems Journal Special Issue on Distributed and Parallel Knowledge Discovery*, 3:422–448, 2001.

[KK96] Daniel A. Keim and Hans-Peter Kriegel. Visualization techniques for mining large databases: A comparison. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):932–938, 1996. Special Issue on Data Mining.

[KK98a] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal of Scientific Computing*, 20(1):359–392, 1998.

[KK98b] G. Karypis and V. Kumar. A parallel algorithm for multilevel graph-partitioning and sparse matrix ordering. *Journal of Parallel and Distributed Computing*, 48(1):71–95, 1998.

[KL70] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49:291–307, 1970.

[KLF01] Branko Kavšek, Nada Lavrač, and Anuška Ferligoj. Consensus decision trees: Using consensus hierarchical clustering for data relabelling and reduction. In *Proceedings of ECML 2001*, volume 2167 of *LNAI*, pages 251–262. Springer, 2001.

[Koh90] Teuvo Kohonen. The self-organizing map. *Proc. IEEE*, 78(9):1464–80, Sept 1990.

[Koh95] Teuvo Kohonen. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).

[Kol97] T. Kolda. *Limited-Memory Matrix Methods with Applications*. PhD thesis, University of Maryland, College Park, 1997.

[KPHJ99] H. Kargupta, B. Park, D. Hershberger, and E. Johnson. Collective data mining: A new perspective toward distributed data mining. In Hillol Kargupta and Philip Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*. MIT/AAAI Press, 1999.

[KR90] L. Kaufmann and P. Rousseeuw. *Finding Groups in Data: an Introdution to Cluster Analysis*. John Wiley and Sons, 1990.

[Kum00] Shailesh Kumar. *Modular Learning Through Output Space Decomposition*. PhD thesis, The University of Texas at Austin, December 2000.

[KV95] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. In D. S. Touretzky G. Tesauro and T. K. Leen, editors, *Advances in Neural Information Processing Systems-7*, pages 231–238, 1995.

[KW99] J. Kim and T. Warnow. Tutorial on phylogenetic tree estimation. In *Intelligent Systems for Molecular Biology, Heidelberg*, 1999.

[KWY95] S. Kannan, T. Warnow, and S. Yooseph. Computing the local consensus of trees. In *Association for Computing Machinery and the Society of Industrial Applied Mathematics, Proceedings, ACM/SIAM Symposium on Discrete Algorithms*, pages 68–77, 1995.

[Lan95] Ken Lang. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*, pages 331–339, 1995.

[Law01] R. D. Lawrence et al. Personalization of supermarket product recommendations. *Data Mining and Knowledge Discovery*, 4(1/2):11–32, 2001.

[Lew92] D. D. Lewis. Feature selection and feature extraction for text categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 212–217, San Mateo, California, February 1992. Morgan Kaufmann.

[MC85] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–179, 1985.

[Meh99] Mala Mehrotra. Multi-viewpoint clustering analysis (MVP-CA) technology for mission rule set development and case-based retrieval. Technical Report AFRL-VS-TR-1999-1029, Air Force Research Laboratory, 1999.

[Mil81] G. W. Milligan. A review of Monte Carlo tests of cluster analysis. *Multivariate Behavioral Research*, 16:379–407, 1981.

[Mir01] Boris Mirkin. Reinterpreting the category utility function. *Machine Learning*, 42(2):219–228, November 2001.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[MJ95] J. Mao and A. K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Trans. on Neural Networks*, 6(2):296–317, March 1995.

[MMK01] Ion Muslea, Steve Minton, and Craig Knoblock. Selective sampling + semi-supervised learning = robust multi-view learning. In *IJCAI-2001 Workshop on Text Learning Beyond Supervision*, 2001.

[MMR97] K. Mehrotra, C. Mohan, and S. Ranka. *Elements of Artificial Neural Networks*. MIT Press, Cambridge, Massachusetts, 1997.

[MN98] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[MR99] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings pf the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation*, pages 195–204, 1999.

[MS00] Dharmendra S. Modha and W. Scott Spangler. Clustering hypertext with applications to web searching. In *Proceedings of the ACM Hypertext 2000 Conference, San Antonio, TX, May 30-June 3*, 2000.

[Mur85] Fionn Murtagh. *Multidimensional Clustering Algorithms*. Physica-Verlag, Heidelberg and Vienna, 1985.

[NG00] K. Nigam and R. Ghani. Analyzing the applicability and effectiveness of co-training. In *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pages 86–93. ACM, 2000.

[NH94] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th VLDB Conference, Santiago Chile*, pages 144–155, 1994.

[Nie81] H. Niemann. *Pattern Analysis*. Springer, Berlin, 1981.

[NMTM98] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 792–799. AAAI Press, 1998.

[NN86a] D. A. Neumann and V. T. Norton. Clustering and isolation in the consensus problem for partitions. *Journal of Classification*, 3:281–298, 1986.

[NN86b] D. A. Neumann and V. T. Norton. On lattice consensus methods. *Journal of Classification*, 3:225–256, 1986.

[PCS00] A. Prodromidis, P. Chan, and S. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*. AAAI/MIT Press, Cambridge, MA, 2000.

[Per93] M. P. Perrone. *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization*. PhD thesis, Brown University, May 1993.

[PPS01] C. Perlich, F. Provost, and J. Simonoff. Tree induction vs. logistic regression: A learning-curve analysis. Technical Report IS-01-02, Stern School of Business, New York University, 2001. CeDER Working Paper.

[Pra94] Lorien Y. Pratt. Experiments on the transfer of knowledge between neural networks. In S. Hanson, G. Drastal, and R. Rivest, editors, *Computational Learning Theory and Natural Learning Systems, Constraints and Prospects*, chapter 19, pages 523–560. MIT Press, 1994.

[Pri94] C. E. Priebe. Adaptive mixtures. *Journal of the American Statistical Association*, 89:796–806, 1994.

[PSL90] A. Pothen, H. Simon, and K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Applications*, 11:430–452, 1990.

[Rag01] Thomas Ragg. Building committees by clustering models based on pairwise similarity values. In *Proc. ECML 2001*, volume 2167 of *LNAI*, pages 406–418. Springer, 2001.

[Ras92] E. Rasmussen. Clustering algorithms. In W. Frakes and R. Baeza-

Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice Hall, New Jersey, 1992.

[RL91] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural Computation*, 3(4):461–483, 1991.

[RS99] Rajeev Rastogi and Kyuseok Shim. Scalable algorithms for mining large databases. In Jiawei Han, editor, *KDD-99 Tutorial Notes*. ACM, 1999.

[SA99] Alexander Strehl and J. K. Aggarwal. Detecting moving objects in airborne forward looking infra-red sequences. In *Proc. IEEE Workshop on Computer Vision Beyond the Visible Spectrum (CVPR 1998), Fort Collins*, pages 3–12. IEEE, June 1999.

[SA00a] Alexander Strehl and J. K. Aggarwal. MODEEP: A motion-based object detection and pose estimation method for airborne FLIR sequences. *Machine Vision and Applications*, 11(6):267–276, 2000.

[SA00b] Alexander Strehl and J. K. Aggarwal. A new Bayesian relaxation framework for the estimation and segmentation of multiple motions. In *Proc. IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI 2000), Austin*, pages 21–25. IEEE, April 2000.

[Sal89] Gerard Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley (Reading MA), 1989.

[SB88] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[SG00a] Alexander Strehl and Joydeep Ghosh. Clustering guidance and quality evaluation using relationship-based visualization. In *Proc. ANNIE 2000, St. Louis*, volume 10, pages 483–488. ASME, November 2000.

[SG00b] Alexander Strehl and Joydeep Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *Proc. HiPC 2000, Bangalore*, volume 1970 of *LNCS*, pages 525–536. Springer, December 2000.

[SG00c] Alexander Strehl and Joydeep Ghosh. Value-based customer grouping from large retail data-sets. In *Proc. SPIE Conference on Data Mining and Knowledge Discovery, Orlando*, volume 4057, pages 33–42. SPIE, April 2000.

[SG01] Alexander Strehl and Joydeep Ghosh. Relationship-based visualization of high-dimensional data clusters. In *Proc. Workshop on Visual Data Mining (KDD 2001), San Francisco*, pages 90–99. ACM, August 2001.

[SG02a] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. In *Proc. AAAI 2002, Edmonton*. AAAI/MIT Press, July 2002. In press.

[SG02b] Alexander Strehl and Joydeep Ghosh. Relationship-based clus-

tering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 2002. In press.

[SGM00] Alexander Strehl, Joydeep Ghosh, and Raymond J. Mooney. Impact of similarity measures on web-page clustering. In *Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin*, pages 58–64. AAAI/MIT Press, July 2000.

[Sha96] A. Sharkey. On combining artificial neural networks. *Connection Science*, 8(3/4):299–314, 1996.

[SKK99] K. Schloegel, G. Karypis, and V. Kumar. Parallel multilevel algorithms for multi-constraint graph partitioning. Technical Report 99-031, Dept of Computer Sc. and Eng, Univ. of Minnesota, 1999.

[SKK00] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[SM96] D. Silver and R. Mercer. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science Special Issue: Transfer in Inductive Systems*, 1996.

[Smy96] Padhraic Smyth. Clustering using Monte Carlo cross-validation. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, OR*, pages 126–133. AAAI Press, 1996.

[Spe06] C. Spearman. Footrule' for measuring correlations. *British Journal of Psychology*, 2:89–108, July 1906.

[SS97] H. Schutze and H. Silverstein. Projections for efficient document clustering. In *Proceedings of SIGIR'97, Philadelphia*, pages 74–81, 1997.

[ST00] N. Slonim and N. Tishby. Agglomerative information bottleneck. In *Proc. of NIPS-12, 1999*, pages 617–623. MIT Press, 2000.

[Str88] Gilbert Strang. *Linear Algebra and its Applications, 3rd edition.* Harcourt Brace Janovich, Orlando, FL, 1988.

[Str98] Alexander Strehl. A new Bayesian relaxation algorithm for motion estimation and segmentation in the presence of two affine motions. Master's thesis, The University of Texas at Austin, August 1998.

[SWY75] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[TG96] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, 1996.

[TG99] K. Tumer and J. Ghosh. Linear and order statistics combiners for pattern classification. In A. Sharkey, editor, *Combining Artificial Neural Nets*, pages 127–162. Springer-Verlag, 1999.

[Thr96] S. Thrun. Explanation-based neural network learning: A lifelong learning approach. *Machine Learning*, 8:323–339, 1996.

[TO96] Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The TC alogorithm. In *The 13th International Conference on Machine Learning*, pages 489–497, 1996.

[Tor52] W. S. Torgerson. Multidimensional scaling, i: theory and method. *Psychometrika*, 17:401–419, 1952.

[TP97] S. Thrun and L. Y. Pratt. *Learning To Learn*. Kluwer Academic, Norwell, MA, 1997.

[Tuf83] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut, 1983.

[Vap95] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[vR79] C. J. van Rijsbergen. *Information Retrieval*. Buttersworth, London, 1979.

[Wat69] S. Watanabe. *Knowing and Guessing - A Formal and Quantative Study*. John Wiley and Sons Inc., 1969.

[Wil88] P. Willet. Recent trends in hierarchical document clustering: a criticial review. *Information Processing and Management*, 24(5):577–597, 1988.

[Wol92] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[Yan99] Y. Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88, May 1999.

[YC74] T. Y. Young and T. W. Calvert. *Classification, Estimation and Pattern Recognition*. Elsevier, New York, 1974.

[YP97] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann, 1997.

[ZE98] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Proceedings of the 21st Annual International ACM SIGIR Conference*, pages 46–54, 1998.

[Zip29] George Kingsley Zipf. Relative frequency as a determinant of phonetic change. *Reprinted from the Harvard Studies in Classical Philiology*, XL, 1929.

[ZK01] Ying Zhao and George Karypis. Criterion functions for document clustering: Experiments and analysis. Technical Report 01-40, University of Minnesota, 2001.

[ZRL97] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.