Relationship-Based Clustering and Visualization for High-Dimensional Data Mining

Alexander Strehl • Joydeep Ghosh
Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, Texas 78712-1084, USA
strehl@ece.utexas.edu • ghosh@ece.utexas.edu

In several real-life data-mining applications, data reside in *very high* (1000 or 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 paper 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 efficient 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 two dimensions, with clusters showing up as bands. While two-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 toward the right number of clusters. Results are presented on a real retail industry dataset of several thousand customers and products, as well as on clustering of web-document collections and of web-log sessions.
(*Cluster Analysis; Graph Partitioning; High Dimensional; Visualization; Retail Customers; Text Mining; Web-Log Analysis*)

1. Introduction

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) (Kohonen 1990) imposes a logical, "topographic" *ordering* on the cluster centers such that centers that are nearby in the logical ordering represent

0899-1499/02/0000/0001$5.00
1526-5528 electronic ISSN

nearby clusters in the feature space. A popular choice for the logical ordering is a two-dimensional lattice that allows all the data points to be projected onto a two-dimensional plane for convenient visualization (Haykin 1999). 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:

1. Grouping customers based on buying behavior to provide useful marketing decision-support knowledge, especially in e-business applications where electronically observed behavioral data are readily available. Customer clusters can be used to identify up-selling and cross-selling opportunities with existing customers (Lawrence et al. 2001).

2. 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: (a) large sample size, n, and (b) 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 paper, but see Gupta and Ghosh (2001) for a solution for retail data.

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 Han et al. (2001), 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 (Chang and Ghosh 2001) means. Extensive approaches for feature selection or extraction have been long studied, particularly in the pattern-recognition community (Young and Calvert 1974, Mao and Jain 1995, Duda et al. 2001). 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 are 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 two- or three-dimensional manifold within the high-dimensional embedding space. The simplest example of case (i) is where the data are 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 (Faloutsos and Lin 1995), can be effectively applied. For further description of these methods, see Section 7 on related work.

This paper 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 (Friedman 1994). We have repeatedly encountered such situations when examining retail industry market-basket data for behavioral customer clustering, and also certain web-based data collection.

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 paper 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 paper applies it to three different domains (i) clustering market-baskets, (ii) web-documents, and (iii) weblogs. 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 it is quadratic, both in computational and storage complexity, with respect to the number of data points, n. This can become problematic for very large databases. Several methods for reducing this complexity are outlined in Section 6, but not elaborated upon much as that is not the primary focus of this present work.

To concretize some of the remarks above and motivate the rest of the paper, let us take a closer look at transactional data. A large market-basket database may involve thousands of customers and productlines. 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 high-dimensional (large number of products), but sparse (most features are zero). Also, transactional data typically have 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). In addition, features are often neither nominal, nor continuous, but have discrete positive ordinal attribute values, with a strongly non-Gaussian distribution.

One way to reduce the feature space is only to consider 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 (Zipf 1929), 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 the 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 only to 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 (Berry and Linoff 1997, Chapter 8). Unfortunately, this results in loss of vital information: One cannot differentiate between buying one gallon of milk and 100 gallons of milk, or one cannot weight importance between buying an apple vs. buying a car, though 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 five-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, 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 paper yield far better results for such transactional data. While the methods have certain properties tailored to such datasets, they can also be applied to other higher-dimensional datasets 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 (Strehl et al. 2001) showed that in this domain too traditional clustering techniques had 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 (McCallum and Nigam 1998) and a normalized version of k-means (Dhillon and Modha 2001) also show decent results. We also apply the technique to clustering visitors to a web-site based 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 2. Section 3 describes a specific clustering technique (OPOSSUM), based on a multi-level graph partitioning algorithm (Karypis and Kumar 1998). In Section 4, we describe a simple but effective visualization technique that is applicable to similarity spaces (CLUSION). Clustering and visualization results are presented in Section 5. In Section 6, we consider system issues and briefly discuss several strategies to scale OPOSSUM for large datasets. Section 7 summarizes related work in clustering, graph partitioning, and visualization.

2. Domain-Specific Features and Similarity Space

2.1. 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 jth column vector representing the sample $\mathbf{x}_j$. $\mathbf{x}_j^t$ 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 one-dimensional SOMs, any top-down recursive bisection approach as well as our proposed method, the labeling contains extra ordering information. Let $\mathscr{C}_\ell$ denote the set of all objects in the ℓth cluster ($\ell \in \{1, \ldots, k\}$), with $\mathbf{x}_j \in \mathscr{C}_\ell \Leftrightarrow \lambda_j = \ell$ and $n_\ell = |\mathscr{C}_\ell|$.

2.2. Process

Figure 1 gives an overview of our relationship-based clustering process from a set of raw object descriptions $\mathscr{X}$ (residing in input space $\mathscr{I}$) via the vector space description $\mathbf{X}$ (in feature space $\mathscr{F}$) and relationship description $\mathbf{S}$ (in similarity space $\mathscr{S}$) to the cluster labels $\boldsymbol{\lambda}$ (in output space $\mathscr{O}$): $(\mathscr{X} \in \mathscr{I}^n) \xrightarrow{\Upsilon} (\mathbf{X} \in \mathscr{F}^n \subset \mathbb{R}^{d \times n}) \xrightarrow{\Psi} (\mathbf{S} \in \mathscr{S}^{n \times n} = [0,1]^{n \times n} \subset \mathbb{R}^{n \times n}) \xrightarrow{\Phi} (\boldsymbol{\lambda} \in \mathscr{O}^n = \{1, \ldots, k\}^n)$. For example in web-page clustering, $\mathscr{X}$ is a collection of n web-pages x_j with $j \in \{1, \ldots, n\}$.

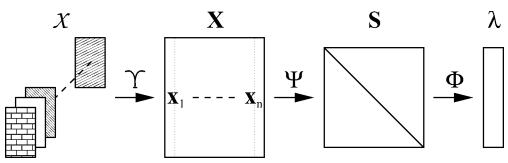

Figure 1 The Relationship-Based Clustering Framework

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 $\boldsymbol{\lambda}$ is computed using a clustering function Φ, such as graph-partitioning. In short, the basic process can be denoted as $\mathscr{X} \xrightarrow{\Upsilon} \mathbf{X} \xrightarrow{\Psi} \mathbf{S} \xrightarrow{\Phi} \boldsymbol{\lambda}$.

2.3. Similarity Measures

In this paper, we work 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 non-zeros 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 latter being preferable due to maximum-likelihood properties (Strehl et al. 2000). Similarity can also be defined by the cosine of the angle between two vectors:

$$s^{(\mathrm{C})}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^\dagger \mathbf{x}_b}{\|\mathbf{x}_a\|_2 \cdot \|\mathbf{x}_b\|_2} \tag{1}$$

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 (Strehl et al. 2000). For binary features, the Jaccard coefficient (Jain and Dubes 1988) 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.

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

The extended Jaccard coefficient is also given by (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 Strehl et al. (2000), where it is shown that the extended Jaccard coefficient is particularly well suited for market-basket data.

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 determine an appropriate measure for the given application, that captures the essential aspects of the class of high-dimensional data distributions being considered.

3. OPOSSUM

In this section, we present 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.

3.1. Balancing

Typically, one segments transactional data into seven to 14 groups, each of which should be of comparable importance. Balancing avoids trivial clusterings (e.g., $k-1$ singletons and one 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 kth 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 one.

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 $\Delta\mathcal{E} = (\mathcal{E} \setminus (\mathcal{E}_1 \cup \mathcal{E}_2 \cup \cdots \cup \mathcal{E}_k))$. The clustering task is thus to find an edge separator with a minimum sum of edge weights, which partitions the graph into k disjoint pieces. The following equation formalizes this *minimum-cut objective*:

$$\min_{\Delta\mathcal{E}} \sum_{(a,b)\in\Delta\mathcal{E}} s(\mathbf{x}_a, \mathbf{x}_b) \tag{3}$$

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

$$k \max_{\ell\in\{1,\ldots,k\}} \sum_{\lambda_j=\ell} w_j \leq t \tag{4}$$

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

Thus, in graph partitioning one has essentially to solve a constrained optimization problem. Finding such an optimal partitioning is an NP-hard problem (Garey and Johnson 1979). 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 (Kernighan and Lin 1970) 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 (Pothen et al. 1990, Hendrickson and Leland 1995) uses the eigenvector associated with the second smallest eigenvalue of the graph's Laplacian (Fiedler vector) (Fiedler 1975) for splitting.

Metis (Karypis and Kumar 1998) handles multi-constraint multi-objective graph partitioning in three phases: coarsening, initial partitioning, and refining. First a sequence of successively smaller and therefore coarser graphs is constructed through heavy-edge matching. Second, 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 swapped only among 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 Section 6.1. In contrast, since recursive bisection computes a bisection of a subgraph at a time, its view is limited. Thus, it cannot 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. Determining the Number of Clusters

Finding the "right" number of clusters k for a dataset 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 in which one is interested. 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 are 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 (Strehl et al. 2000).

For transactional data, the number is often specified by the end-user to be typically between seven and 14 (Berry and Linoff 1997). Otherwise, one can employ a suitable heuristic to obtain an appropriate value of k during the clustering process. This section describes how we find a desirable clustering, with *high overall cluster quality* $\phi^{(Q)}$ and a *small number of clusters* k. Our objective is to maximize intra-cluster similarity and minimize inter-cluster similarity, given by $\text{intra}(\mathbf{X}, \boldsymbol{\lambda}, i) = \frac{2}{(n_i-1)\cdot n_i} \sum_{\lambda_a=\lambda_b=i,\, b>a} s(\mathbf{x}_a, \mathbf{x}_b)$ and $\text{inter}(\mathbf{X}, \boldsymbol{\lambda}, i, j) = \frac{1}{n_i \cdot n_j} \sum_{\lambda_a=i,\, \lambda_b=j} s(\mathbf{x}_a, \mathbf{x}_b)$, 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 $\phi^{(Q)} \in [0, 1]$ ($\phi^{(Q)} < 0$ in case of pathological/inverse clustering) based on the ratio of weighted average inter-cluster to weighted average intra-cluster similarity:

$$\phi^{(Q)}(\mathbf{X}, \boldsymbol{\lambda}) = 1 - \frac{\sum_{i=1}^{k} \frac{n_i}{n-n_i} \sum_{j \in \{1,\ldots,i-1,i+1,\ldots,k\}} n_j \cdot \text{inter}(\mathbf{X}, \boldsymbol{\lambda}, i, j)}{\sum_{i=1}^{k} n_i \cdot \text{intra}(\mathbf{X}, \boldsymbol{\lambda}, i)} \quad (5)$$

$\phi^{(Q)} = 0$ indicates that samples within the same cluster are on average not more similar than samples from different clusters. On the contrary, $\phi^{(Q)} = 1$ describes a clustering where every pair of samples from different clusters has the similarity of zero 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 $\phi^{(Q)}$ as well as a low k, the target function $\phi^{(T)} \in [0, 1]$ is the product of the quality $\phi^{(Q)}$ and a penalty term that 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 $\phi^{(T)}$ and is defined as $\phi^{(T)}(k) = (1 - \frac{2k}{n})\phi^{(Q)}(k)$. 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 first to over-cluster and then use the visualization aid to combine clusters as needed (Section 5.2).

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 5.

4.1. Coarse Seriation

When data are limited to two or three 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 $\boldsymbol{\lambda}$, 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 an $n \times n$ permutation matrix $\mathbf{P}$. The entries $p_{i,j}$ of $\mathbf{P}$ are defined as follows:

$$p_{i,j} = \begin{cases} 1 & \text{if } j = \sum_{a=1}^{i} l_{a,\lambda_i} + \sum_{\ell=1}^{\lambda_i - 1} n_\ell \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The definition of an entry $p_{i,j}$ is based on the entries $l_{i,j}$ of a binary matrix representation of the cluster label vector $\boldsymbol{\lambda}$. The binary $n \times k$ cluster membership indicator matrix $\mathbf{L}$ is defined by each entry $l_{i,j}$:

$$l_{i,j} = \begin{cases} 1 & \text{if } \lambda_i = j \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

In other words, $p_{i,j}$ is 1 if j is the sum of the number of points among 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 $\boldsymbol{\lambda}'$ and data matrix $\mathbf{X}'$ are:

$$\mathbf{S}' = \mathbf{P}\mathbf{S}\mathbf{P}^\dagger, \quad \boldsymbol{\lambda}' = \mathbf{P}\boldsymbol{\lambda}, \quad \mathbf{X}' = \mathbf{P}\mathbf{X} \tag{8}$$

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 (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 (Murtagh 1985, Eisen et al. 1998).

4.2. Visualization

The seriation of the similarity matrix, $\mathbf{S}'$, is very useful for visualization. Since the similarity matrix is two-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 black horizontal and vertical lines are used to show the divisions into the rectangular regions. Visualizing similarity space in this way can help to get a feel quickly for the clusters in the data. Even for a large number of points, a sense for the intrinsic number of clusters k in a dataset can be gained.

Figure 2 shows CLUSION output in four extreme scenarios to provide a feel for how data properties

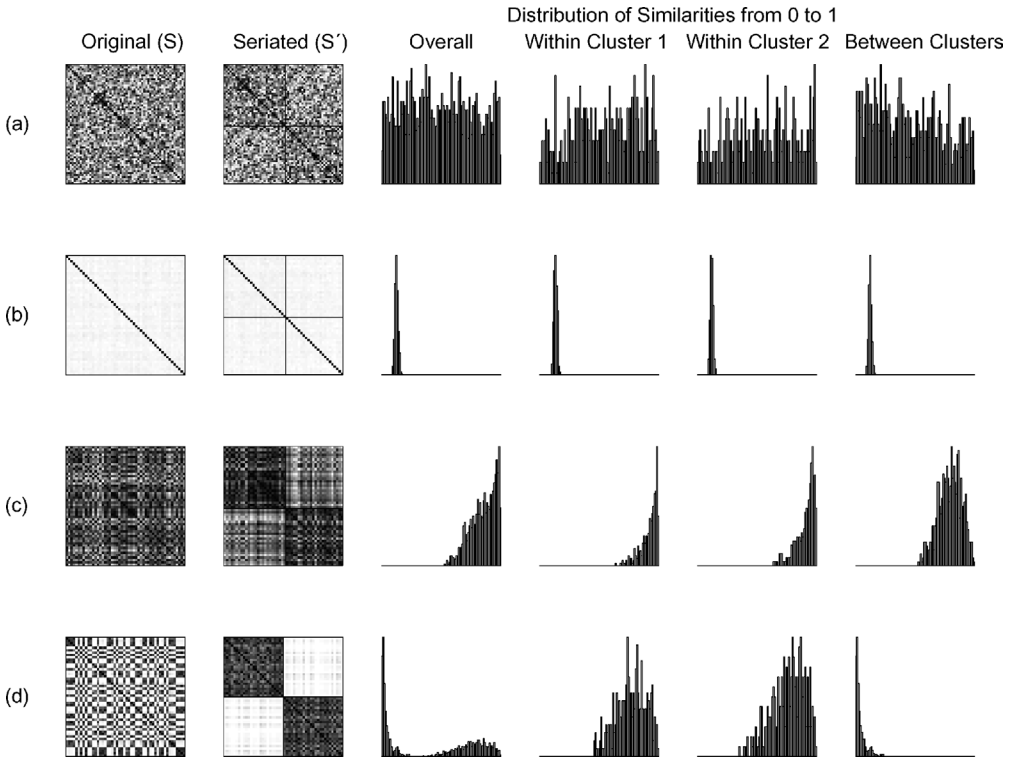

Figure 2 **Illustrative CLUSION Patterns in Original Order and Seriated Using Optimal Bipartitioning Are Shown in the Left Two Columns. The Right Four Columns Show Corresponding Similarity Distributions. In Each Example There Are 50 Objects: (a) No Natural Clusters (Randomly Related Objects), (b) Set of Singletons (Pairwise Near Orthogonal Objects), (c) One Natural Cluster (Unimodal Gaussian), (d) Two Natural Clusters (Mixture of Two Gaussians)**

translate to the visual display. Without loss of generality, we consider the partitioning of a set of objects into two 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 are 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 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 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 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 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 these data are actually one large cluster. In Figure 2(d), the gray-level distribution of $\mathbf{S}$

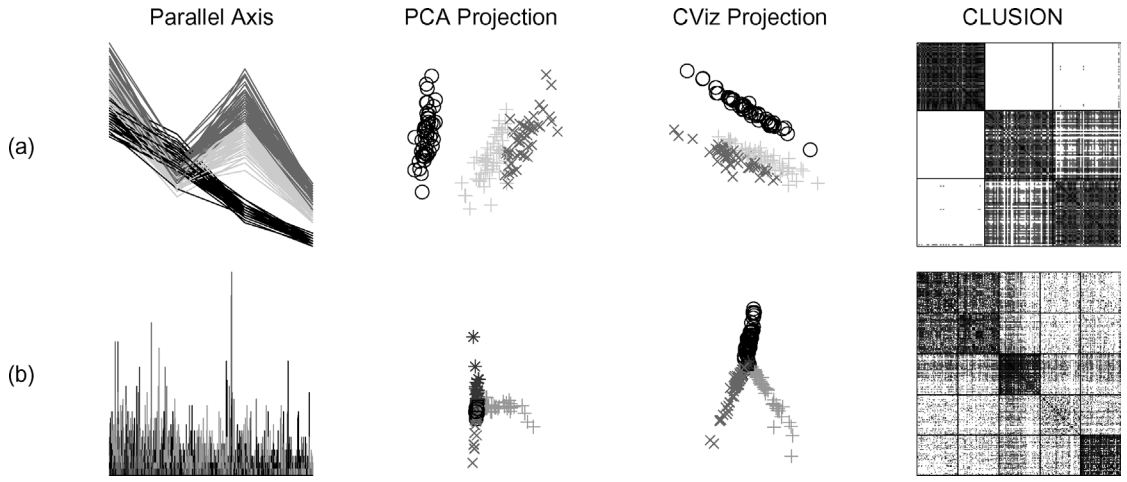

Figure 3 **Comparison of Cluster-Visualization Techniques. All Tools Work Well on the Four-Dimensional IRIS Data (a). But on the 2903-Dimensional Yahoo! News-Document Data (b), Only CLUSION Reveals that Clusters 1 and 2 Are Actually Highly Related, Cluster 3 Is Strong and Interdisciplinary, 4 Is Weak, and 5 Is Strong**

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. A too small value of k leaves the on-diagonal regions heterogeneous. On the contrary, growing k beyond the natural number of clusters will introduce dark off-diagonal regions. Finally, CLUSION can be used to compare visually the appropriateness of different similarity measures. Let us assume, for example, that each row in Figure 2 illustrates a particular way of defining similarity for the same dataset. Then, CLUSION makes visually apparent that the similarity measure in (d) lends itself much better for clustering than do the measures illustrated in rows (a), (b), and (c).

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 *not* visualized; instead, all pair-wise relationships, the relevant aspect for the purpose of clustering, are displayed.

Figure 3 compares CLUSION with some other popular visualizations. In Figure 3(a) the 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. Membership in cluster 1/2/3 is indicated by black/dark gray/light gray (parallel axis), black/dark gray/light gray 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(b) shows the same comparison for 293 documents from which 2903 word frequencies were extracted to be used as features. In fact, these data consist of five clusters selected from 40 clusters extracted from a Yahoo! news-document collection that will be described in more detail in Section 5.2. Extra gray shades 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. For example, cluster 4 can hardly be seen in

the PCA projection and CViz. 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 available only where a supervised categorization is given. Of course, the categorization cannot be used to tune the clustering. Clusters 1 and 2 contain only documents from the Health category so they are highly related. The fourth 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 that cluster 3 is not only strong, as indicated by the dark diagonal region, but also has distinctly above-average relationships to *all* the other four 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 dataset, which is of great help to speed up the iterations of analysis during a data-mining project.

5. Experiments

5.1. Retail Market-Basket Clusters

First, we will show clusters in a real retail transaction database of 21672 customers of a drugstore (provided by Knowledge Discovery 1). For the illustrative purpose of this paper, 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 total and were dropped. The remaining $n = 2466$ customers (34 customers had empty baskets after removing the irrelevant products) with their $d = 762$ features 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, a reasonable value of k can be obtained using heuristics (Section 3.3).

OPOSSUM's results for this example were 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). Figure 4 shows the extended Jaccard similarity matrix (83% sparse) using CLUSION in six scenarious: (a) original (randomly) ordered matrix, (b) seriated using Euclidean k-means, (c) using SOM, (d) using standard Jaccard k-means, (e) using extended Jaccard sample balanced OPOSSUM, and (f) using value balanced OPOSSUM clustering. Customer and revenue ranges are given below each image. In (a), (b), (c), and (d) clusters are neither compact nor balanced. In (e) and (f) clusters are much more compact, even though there is the additional constraint that they be balanced, based on an

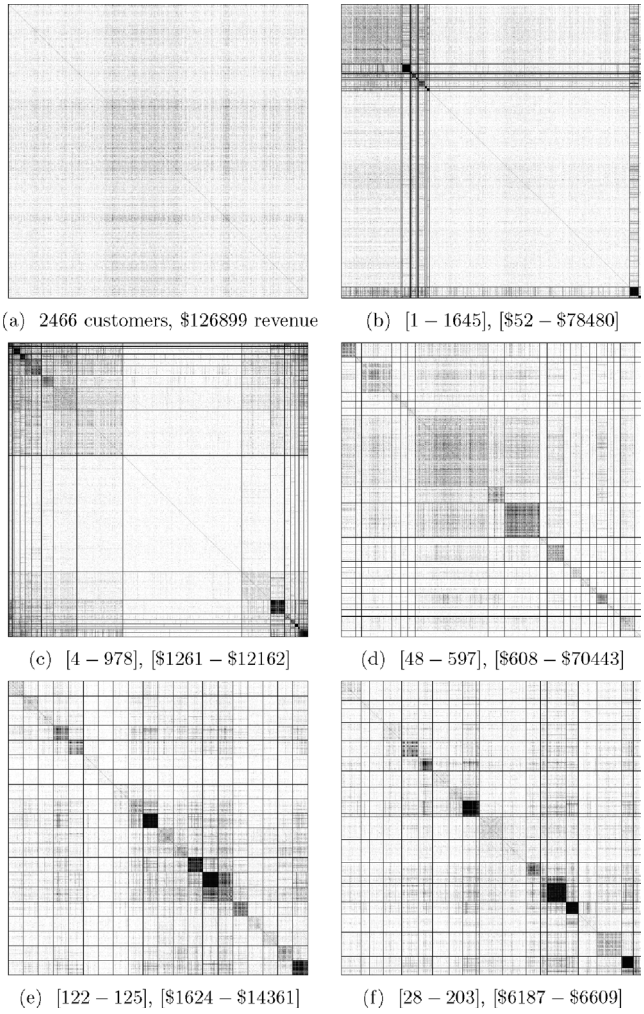

Figure 4 **Visualizing Partitioning Drugstore Customers into 20 Clusters. Relationship Visualizations Using CLUSION: (a) Original (Randomly) Ordered Similarity Matrix, (b) Partially Reordered Using Euclidean *k*-means, (c) Using SOM, (d) Using Standard Jaccard *k*-means, (e) Using Extended Jaccard Sample Balanced OPOSSUM, (f) Using Value Balanced OPOSSUM Clustering. Customer and Revenue Ranges Are Given Below Each Image**

equal number of customers and equal revenue metrics, respectively. Below each CLUSION visualization, the ranges of numbers of customers and revenue totals in $ among the 20 clusters are given to indicate balancedness. We also experimented with minimum-distance agglomerative clustering but this resulted in 19 singletons and one 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 4(b)). The SOM after 100000 epochs performs slightly better (Figure 4(c)) but is outperformed by the standard Jaccard *k*-means (Figure 4(d)) which is adopted to similarity space by using $\sqrt{-\log(s^{(J)})}$ as distances (Strehl et al. 2000). As the relationship-based CLUSION shows, OPOSSUM (Figure 4(e), (f)) 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 in a representative solution. (The solution for *k*-means depends on the initial choices for the means.) 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 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 dataset.

In Table 1 the top three descriptive and discriminative products for the customers in the 20 value balanced clusters are shown (see also Figure 4(f)). Customers in cluster $\mathscr{C}_2$, 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 $\mathscr{C}_{15}$, which can be seen to be a highly compact cluster of Christmas shoppers characterized by greeting-card and candy purchases. Note that OPOSSUM had an

Table 1 **List of *Descriptive* (top) and *Discriminative* Products (bottom) Dominant in Each of the 20 Value Balanced Clusters Obtained From the Drugstore Data (see also Figure 4(f)). For Each Item the Average Amount of $ Spent in This Cluster and the Corresponding Lift is Given. Product Names are Partially Abbreviated in the Original Data**

$\mathcal{C}_\ell$	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

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 1(top)), cluster $\mathcal{C}_9$ are the moderate spenders and cluster $\mathcal{C}_{10}$ are the big spenders, as cluster $\mathcal{C}_{10}$ is much smaller with equal revenue contribution (Figure 4(f)). Our hunch is reinforced by looking at Figure 4(f).

5.2. Web-Document Clusters

In this Section, we present results on documents from the Yahoo! news section. Each of the 2340 documents is characterized by a bag of words. The data are publicly available from `ftp://ftp.cs.umn.edu/dept/users/boley/PDDPdata/` (K1 series) and was used in Boley et al. (1999) and Strehl et al. (2000). 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 $1, \ldots, 20$, respectively. The raw 21839×2340 word-by-document matrix consists of the non-normalized occurrence frequencies of stemmed words, using Porter's suffix stripping algorithm (Frakes 1992). 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 $d = 2903$.

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! news dataset 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 5 shows clusterings with three settings of k. For $k = 10$ (Figure 5(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 k to 40 (Figure 5(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 5(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.

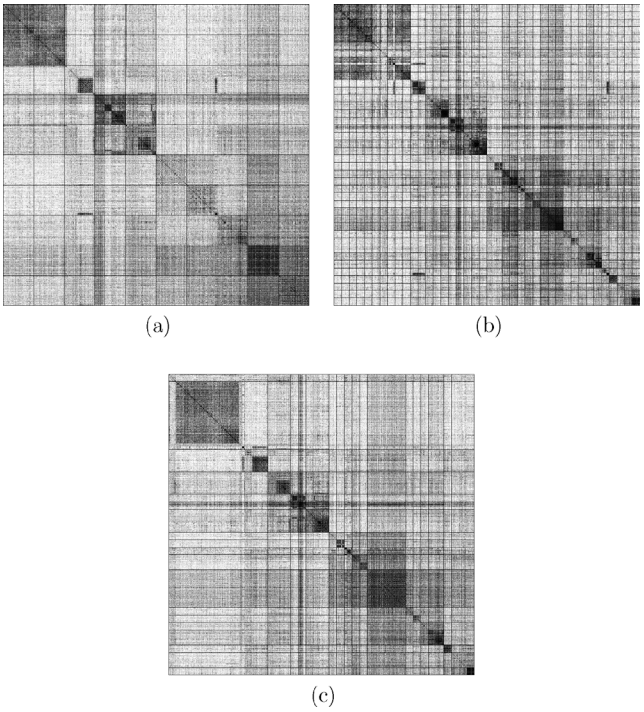

Figure 5 **Comparison of Various Number of Clusters k for Yahoo! News Data: (a) Under-Clustering at $k = 10$, (b) Over-Clustering at $k = 40$, (c) Good Clustering Through Interactive Split and Merge Using CLUSION at $k = 20$**

Table 2(top) shows cluster evaluations, and their descriptive and discriminative word stems. Each cluster ($\mathscr{C}_\ell$) is evaluated using the dominant category ($\mathscr{K}_{\hat{h}}$), purity ($\phi^{(P)}$), and entropy ($\phi^{(E)}$). Let $n_\ell^{(h)}$ denote the number of objects in cluster $\mathscr{C}_\ell$ that are classified to be in category h as given by the original Yahoo! categorization. Cluster $\mathscr{C}_\ell$'s *purity* can be defined as

$$\phi^{(P)}(\mathscr{C}_\ell) = \frac{1}{n_\ell} \max_h \left(n_\ell^{(h)}\right). \tag{9}$$

Purity can be interpreted as the classification rate under the assumption that all samples of a cluster are predicted to be members of the actual dominant class for that cluster. Alternatively, we also use

Table 2 Cluster Evaluations, Their Descriptive and Discriminative Terms (top) as Well as the Confusion Matrix (bottom) for the Yahoo! News Example (see also Figure 5(c)). For Each Cluster Number $\mathscr{C}_\ell$ the Dominant Category $\mathscr{H}_{\hat{h}}$, Purity $\phi^{(P)}$, and Entropy $\phi^{(E)}$ Are Shown

$\mathscr{C}_\ell$	$\mathscr{H}_{\hat{h}}$	$\phi^{(P)}$	$\phi^{(E)}$	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**

$[0, 1]$ *entropy*, which is defined for a problem with g categories as

$$\phi^{(E)}(\mathscr{C}_\ell) = -\sum_{h=1}^{g} \frac{n_\ell^{(h)}}{n_\ell} \log_g \left(\frac{n_\ell^{(h)}}{n_\ell} \right). \qquad (10)$$

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 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 considered. The evaluation metrics serve the purpose of validating our results and 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 $\mathscr{C}_2$ 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).

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) was suggested (Banerjee and Ghosh 2001) to describe how similar two trajectories are. 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, where entries in the data matrix $\mathbf{X}$ indicate time spent in a particular session (column) on a particular page (row).

In this Section, we present results of OPOSSUM and CLUSION for the data presented in Banerjee and Ghosh (2001). 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 6 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 regions, showing a better and more balanced grouping. The cosine similarity is far less sparse and is dominated by major category index pages, while the LCS shows better isolation among the clusters. 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 6(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.

6. System Issues

6.1. Synergy Between OPOSSUM and CLUSION

The visualization and clustering techniques presented in this work need to be considered together, 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 linear time ($O(n)$) 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 $\mathscr{C}_1$ and $\mathscr{C}_2$ appear contiguous in Figure 5(a). Finally, one can experiment with different similarity measures for OPOSSUM and quickly get visual feedback regarding their effectiveness using CLUSION (Figure 6).

6.2. Scalability

The computational bottleneck in the overall process lies in calculating the similarity matrix, which involves $O(n^2d)$ operations, since similarity needs to be computed between each pair of data points, and involves all the dimensions. By exploiting sparsity, computation of a single similarity value can be reduced from $O(d)$ to O(number of non-zeros in d). However, once this matrix is computed, any subsequent clustering routine does not depend on d at all! Metis is very fast, almost linear in the number of vertices for reasonably sparse graphs, as has been shown over numerous experiments (Karypis and Kumar 1998). Finally, the reordering of the similarity matrix for visualization is $O(n)$. Thus the overall method is linear in d.

The quadratic complexity w.r.t. the number of objects, n, is problematic for large datasets. Note that any clustering algorithm that compares an object with all others (e.g., agglomerative, all relationship-based methods) has a complexity at least $O(n^2)$, as does OPOSSUM. There are four main ways of reducing this computation. We mention them briefly and then explore the first option in a bit more detail.

1. 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 (Cutting et al. 1992). 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 (Indyk 1999). We will 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.

2. 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 (Zhang et al. 1997). 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 (Ben-Dor et al. 1999) 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, Bradley et al. 1998) in main memory. Subsequent refinement based on summarized information is then restricted to main-memory operations without resorting to further disk scans.

3. 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,

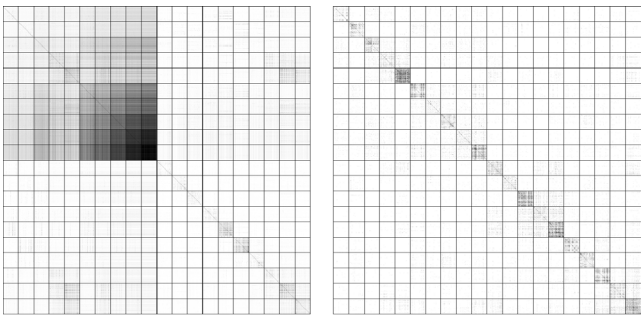

Figure 6 **Web-Log Session Clustering Using a Vector-Space Model and Cosine Similarity (a), and Using LCS Similarity (b)**

become sensitive to the initial selection of representatives. Also, representatives might have to be updated resulting in an iterative algorithm.

4. 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 pre-processing 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 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 (including OPOSSUM) to save computation, at the cost of some loss in quality.

6.3. 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 a 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 a 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 (Gusfield and Irving 1989), but this will increase the computation time.

Figure 7 illustrates the behavior of FASTOPOSSUM for the drugstore customer dataset from Section 5.1. 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 (Figure 7(a)), however the imbalance increases (Figure 7(b)). The remaining edge-weight fraction indicates how much of the cumulative edge weight remains after the edge separator has been removed:

$$\left(\sum_{\ell=1}^{k}\sum_{\lambda_a=\ell}\sum_{\lambda_b=\ell,\, b>a} s(\mathbf{x}_a, \mathbf{x}_b)\right)\Big/\left(\sum_{a=1}^{n}\sum_{b=a+1}^{n} s(\mathbf{x}_a, \mathbf{x}_b)\right).$$

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 significantly decreased quality in terms of remaining edge weight (Figure 7(a)). However, the balancing property is progressively relaxed as the boot sample becomes smaller in comparison to the full dataset (Figure 7(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.

6.4. Parallel Implementation

Another notion of scalability is w.r.t. the number of processors (speedup, iso-efficiency, etc.). Our analysis (Strehl and Ghosh 2000) shows almost linear speedup

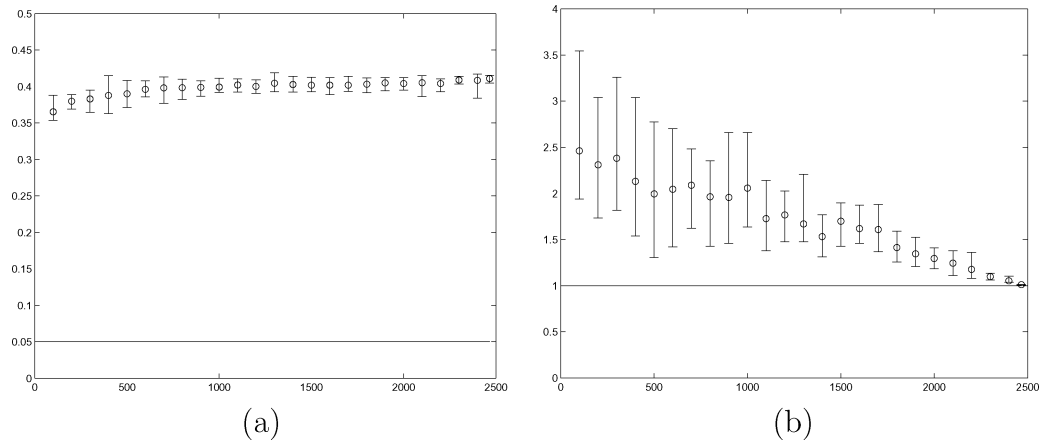

Figure 7 **Effect of Sub-Sampling on OPOSSUM. Cluster Quality As Measured by Remaining Edge-Weight Fraction (a) and Imbalance (b) of *Total* Graph With 2466 Vertices (customers from Section 5.1) for Various Boot Sample Sizes n in FASTOPOSSUM. For Each Setting of n the Results' Range and Mean of 10 Trials Are Depicted**

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 (Schloegel et al. 1999) 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 Strehl and Ghosh (2000).

7. Related Work

7.1. Clustering and Indexing

Clustering has been widely studied in several disciplines, specially since the late 1960s (Jain and Dubes 1988, Hartigan 1975). Classic approaches include partitional methods such as k-means and k-medioids, bottom-up hierarchical approaches such as single link or complete link agglomerative clustering (Murtagh 1983), soft-partitioning approaches such as fuzzy clustering, EM-based techniques and methods motivated by statistical mechanics (Chakaravathy and Ghosh 1996). While several methods of clustering data defined by pairwise (dis)similarities are available (Kaufmann and Rousseeuw 1990), most classical techniques, as well as recent techniques proposed in the data-mining community (CLARANS, DBScan, BIRCH, CLIQUE, CURE, WaveCluster etc, Rastogi and Shim 1999), are based on distances between the samples in the original feature space. The emphasis of the data-mining-oriented proposals mentioned above is primarily on an efficient and scalable (w.r.t. number of records) implementation of approximate k-means, k-medioids, or local density estimation. Thus they are all faced with the "curse of dimensionality" (Friedman 1994) and the associated sparsity issues, when dealing with very high-dimensional data. 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 affect any method based on spatial density, unless the data follow certain simple distributions as described in the introduction. Certain other limitations of popular clustering methods are nicely illustrated in (Karypis et al. 1999). In Aggarwal (2001), the authors recognize that one way of tackling high-dimensional data is to change the distance function in an application-specific way. They suggest some possible modified functions

and principles but do not provide any experimental results.

In databases, where clustering is often tied to the need for efficient indexing, a variety of space-partitioning methods (e.g. R-trees and variants) and data-partitioning (such as KDB-trees), exist. These methods are typically tractable for up to 10 to 15-dimensional data, and by a judicious hybrid of these two approaches, data with tens of attributes may be partitioned (Chakrabarti and Mehrotra 1999). Significant overlaps among the hyper-rectangles and the occurrences of several empty areas become increasingly problematic in the dimensionality is further increased (see Chakrabarti and Mehrotra 1999 for more details).

Graph-theoretic clustering has been known for a while (Jain and Dubes 1988) though not commonly applied. But lately, such an approach has proved attractive for gene-expression analysis (Ben-Bor et al. 1999).

Graphical methods also have emerged in the data-mining literature to tackle high-dimensional data analysis. ROCK (Robust Clustering using linKs, Guha et al. 1999) is an agglomerative hierarchical clustering technique for categorical attributes. It uses the binary Jaccard coefficient and a thresholding criterion to establish links between samples. Common neighbors are used to define inter-connectivity of clusters that is used to merge clusters. CHAMELEON (Karypis et al. 1999) 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 relative inter-connectivity and relative closeness measures. These localized measures lead to a dynamic adaption capability with spectacular results for two-dimensional data. But its effectiveness and interpretability for higher-dimensional data is not reported. In Han et al. (1998), a hypergraph-clustering approach was taken for clustering highly related items defined in high-dimensional space, and generates the corresponding association rules. This method was applied to binarized data, with each frequent item-set being represented by a weighted hyperedge. Like our method, it is suitable for high-dimensional data and is linear in d. Subsequently, this and another graph-partitioning algorithm called principal direction divisive partitioning was applied for web-document categorization (Boley et al. 1999). These two algorithms are the closest in spirit to our approach.

Finally, spectral partitioning methods (Pothen et al. 1990, Miller et al. 1997) can be applied to similarity graphs. A probabilistic foundation for spectral methods for clustering and segmentation has been recently proposed (Meila and Shi 2001).

Related work on scalability issues of clustering are discussed in Section 6.2.

7.2. Visualization

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

1. Dimensionality reduction by selection of two or three dimensions, or, more generally, projecting the data down to two or three dimensions. Often these dimensions correspond to principal components or a scalable approximation thereof (e.g., FASTMAP, Faloutsos and Lin 1995). Chen (1999), for example, creates a browsable 2-dimensional space of authors through co-citations. Another noteworthy method is CViz (Dhillon et al. 1998), which projects onto the plane that passes through three selected cluster centroids to yield a "discrimination optimal" two-dimensional projection. These projections are useful for a medium number of dimensions, i.e., if d is not too large ($<$100). For text mining, linearly projecting down to about 20–50 dimensions does not affect results much (e.g. latent semantic indexing). However, it is still too high to visualize. A projection to lower dimensions leads to substantial degradation and three-dimensional projections are of very limited utility. Nonlinear projections have also been studied (Chang and Ghosh 2001). Recreating a two- or three-dimensional space from a similarity graph can also be done through multi-dimensional scaling (Torgerson 1952).

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

3. Kohonen's (1990) Self Organizing Map (SOM) provides an innovative and powerful way of clustering while enforcing constraints on a logical topology

imposed on the cluster centers. If this topology is two-dimensional, one can readily "visualize" the clustering of data. Essentially a two-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 the dimensionality of the feature space, unless the data are largely confined to a much lower order manifold within this space (Chang and Ghosh 2001). Multi-dimensional scaling (MDS) and associated methods also face similar issues.

Our visualization technique involves a smart reordering of the similarity matrix. Ordering of data points for visualization has previously been used in conjunction with clustering in different contexts. For example, in OPTICS (Ankerst et al. 1999), instead of producing an explicit clustering, an augmented ordering of the database is produced. Subsequently, this ordering is used to display various metrics such as reachability values. In cluster analysis of genome data (Eisen et al. 1998) 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 reorderings have also been considered for browsing hypertext (Berry et al. 1996).

A useful survey of visualization methods for data mining in general can be found in Keim and Kriegel (1996). The popular book by Tufte (1983) on visualizing information is also recommended.

8. Concluding Remarks

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 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 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 assessing 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://www.strehl.com/`.

This work originally stemmed from our encounter with several retail datasets, 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, which are currently very popular for classification problems (Vapnik 1995, Joachims 1998). 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 investigate further this connection for a variety of applications (Jaakkola and Haussler 1999).

The clustering algorithm presented in this paper 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.

Acknowledgments

We want to express our gratitude to Mark Davis of Knowledge Discovery 1 (since then acquired by Net Perceptions) for providing the drugstore retail dataset. We also thank Arindam Banerjee for processing the web-log data into web sessions. This research was supported in part by the NSF under Grant ECS-9900353, by an IBM Faculty Fellowship Award, and by Knowledge Discovery 1, Dell, and Intel.

References

Aggarwal, C. 2001. Re-designing distance functions and distance-based applications for high dimensional data. *SIGMOD Record* **30** 13–18.

Ankerst, M., M. M. Breunig, H.-P. Kriegel, J. Sander. 1999. OPTICS: Ordering Points to Identify the Clustering Structure. *Proceedings of the ACM SIGMOD International Conference on Management of Data, Philadelphia, Pennsylvania, USA*. 49–60.

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

Ben-Dor, A., R. Shamir, Z. Yakhini. 1999. Clustering gene expression patterns. *Journal of Computational Biology*, **6** 281–297.

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

Berry, M. W., B. Hendrickson, P. Raghavan. 1996. Sparse matrix reordering schemes for browsing hypertext. *Lectures in Applied Mathematics (LAM)* **32** 99–123. American Mathematical Society, Providence, RI.

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

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

Chakaravathy, S. V., J. Ghosh. 1996. Scale based clustering using a radial basis function network. *IEEE Transactions on Neural Networks* **2** 1250–61.

Chakrabarti, K., S. Mehrotra. 1999. The hybrid tree: An index structure for high dimensional feature spaces. *ICDE*. 440–447.

Chang, K., J. Ghosh. 2001. A unified model for probabilistic principal surfaces. *IEEE Trans. PAMI* **23** 22–41.

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

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

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

Dhillon, I. S., D. S. Modha, W. S. Spangler. 1998. Visualizing class structure of multidimensional data. S. Weisberg, ed. *Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics, Minneapolis, MN, May 13–16 1998*.

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

Eisen, M. B., P. T. Spellman, P. O. Brown, D. Botstein. 1998. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA* 95 14863–14868.

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

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

Frakes, W. 1992. Stemming algorithms. W. Frakes, R. Baeza-Yates, eds. *Information Retrieval: Data Structures and Algorithms*. 131–160. Prentice Hall, Englewood Cliffs, NJ.

Friedman, J. H. 1994. An overview of computational learning and function approximation. V. Cherkassky, J. Friedman, H. Wechsler, eds. *From Statistics to Neural Networks, Proc. NATO/ASI Workshop*. 1–61. Springer Verlag.

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, San Francisco, CA.

Guha, S., R. Rastogi, K. Shim. 1999. Rock: A robust clustering algorithm for categorical attributes. *Proceedings of the 15th International Conference on Data Engineering*. 512–521.

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

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

Han, E.-H., G. Karypis, V. Kumar, B. Mobasher. 1998. Hypergraph based clustering in high-dimensional data sets: A summary of results. *Data Engineering Bulletin* **21** 15–22.

Han, J., M. Kamber, A. K. H. Tung. 2001. Spatial clustering methods in data mining: A survey. H. Miller and J. Hun, eds. *Geographic Data Mining and Knowledge Discovery*. Taylor & Francis, London.

Hartigan, J. A. 1975. *Clustering Algorithms*. Wiley, New York.

Haykin, S. 1999. *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ.

Hendrickson, B., R. Leland. 1995. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing* **16** 452–469.

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

Jaakkola, T. S., D. Haussler. 1999. Exploiting generative models in discriminative classifiers. *Advances in Neural Information Processing Systems* **11** 487–493. MIT Press, Cambridge, MA.

Jain, A. K., R. C. Dubes. 1988. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ.

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

Karypis, G., E.-H. Han, V. Kumar. 1999. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer* **32** 68–75.

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

Kaufmann, L., P. Rousseeuw. 1990. *Finding Groups in Data: An Introdution to Cluster Analysis*. John Wiley and Sons, New York.

Keim, D. A., H.-P. Kriegel. 1996. Visualization techniques for mining large databases: a comparison. *IEEE Transactions on Knowledge and Data Engineering* **8** 932–938. Special Issue on Data Mining.

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

Kohonen, T. 1990. The self-organizing map. *Proc. IEEE* **78** 1464–1480.

Lawrence, R. D., G. S. Almasi, V. Kotlyar, M. S. Viveros, S. S. Duri. 2001. Personalization of supermarket product recommendations. *Data Mining and Knowledge Discovery* **4** 11–32.

Mao, J., A. K. Jain. 1995. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Trans. on Neural Networks* **6** 296–317.

McCallum, A., K. Nigam. 1998. A comparison of event models for naive bayes text classification.

Meila, M., J. Shi. 2001. Learning segmentation by random walks. T. K. Leen, T. G. Dietterich, V. Tresp, eds. *Advances in Neural Information Processing Systems* **13** 873–879. MIT Press, Cambridge, MA.

Miller, G. L., S.-H. Teng, W. Thurston, S. A. Vavasis. 1997. Separators for sphere packings and nearest neighbor graphs. *Journal of the ACM* **44** 1–29.

Murtagh, F. 1983. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal* **26** 354–359.

Murtagh, F. 1985. *Multidimensional Clustering Algorithms*. Physica-Verlag, Heidelberg, Germany and Vienna, Austria.

Pothen, A., H. Simon, K. Liou. 1990. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Applications* **11** 430–452.

Rastogi, R., K. Shim. 1999. Scalable algorithms for mining large databases. J. Han, ed. *KDD-99 Tutorial Notes*. ACM, New York.

Schloegel, K., G. Karypis, V. Kumar. 1999. Parallel multilevel algorithms for multi-constraint graph partitioning. Technical Report 99-031, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN.

Strehl, A., J. Ghosh. 2000. A scalable approach to balanced, high-dimensional clustering of market-baskets. *Proc. HiPC 2000, Bangalore*, *LNCS* **1970** 525–536. Springer, New York.

Strehl, A., J. Ghosh, R. J. Mooney. 2000. Impact of similarity measures on web-page clustering. *Proc. AAAI Workshop on AI for Web Search (AAAI 2000)*, Austin, TX. 58–64. AAAI/MIT Press.

Torgerson, W. S. 1952. Multidimensional scaling, i: theory and method. *Psychometrika*, **17** 401–419.

Tufte, E. R. 1983. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory*. Springer, New York.

Young, T. Y., T. W. Calvert. 1974. *Classification, Estimation and Pattern Recognition*. Elsevier, New York.

Zhang, T., R. Ramakrishnan, M. Livny. 1997. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery* **1** 141–182.

Zipf, G. K. 1929. Relative frequency as a determinant of phonetic change. *Reprinted from the Harvard Studies in Classical Philiology*, XL.

Accepted by Amit Basu; received February 2001; revised August 2001, January 2002; accepted June 2002.