A Scalable Approach to Balanced, High-dimensional Clustering of Market-baskets

Alexander Strehl and Joydeep Ghosh

The University of Texas at Austin,
Austin, TX 78712-1084, USA
{strehl,ghosh}@lans.ece.utexas.edu
http://www.lans.ece.utexas.edu

Abstract This paper presents OPOSSUM, a novel similarity-based clustering approach based on constrained, weighted graph-partitioning. OPOSSUM is particularly attuned to real-life market baskets, characterized by very high-dimensional, highly sparse customer-product matrices with positive ordinal attribute values and significant amount of outliers. Since it is built on top of Metis, a well-known and highly efficient graph-partitioning algorithm, it inherits the scalable and easily parallelizeable attributes of the latter algorithm. Results are presented on a real retail industry data-set of several thousand customers and products, with the help of CLUSION, a cluster visualization tool.

1 Introduction

A key step in market-basket analysis is to cluster customers into relatively homogeneous groups according to their purchasing behavior. A large market-basket database may involve millions of customers and several thousand product-lines. For each product a customer could potentially buy, a feature (attribute) is recorded in the data-set. A feature usually corresponds to some property (e.g., quantity, price, profit) of the goods purchased. Most customers only buy a small subset of products. If we consider each customer as a multi-dimensional data point and then try to cluster customers based on their buying behavior, the problem differs from classic clustering scenarios in several ways [1]:

- *High dimensionality:* The number of features is very high and may even exceed the number of samples. So one has to face with the curse of dimensionality.
- *Sparsity:* Most features are zero for most samples. This strongly affects the behavior of similarity measures and the computational complexity.
- *Significant outliers:* Outliers such as a few, big corporate customers that appear in an otherwise small retail customer data, may totally offset results. 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. Moreover, since the number of features is very high, normalization can become difficult. Due to these issues, traditional clustering techniques work poorly on real-life market-basket data. This paper describes OPOSSUM, a graph-partitioning approach using value-based features, that is well suited for market-basket clustering scenarios, exhibiting some or all of the characteristics described above. We also examine how this approach scales to large data sets and how it can be parallelized on distributed memory machines.

2 Related Work

Clustering has been widely studied in several disciplines, specially since the early 60's [2,3]. Some classic approaches include partitional methods such as k-means, hierarchical agglomerative clustering, unsupervised Bayes, and soft, statistical mechanics based techniques. Most classical techniques, and even fairly recent ones proposed in the data mining community (CLARANS, DBSCAN, BIRCH, CLIQUE, CURE, WAVECLUSTER etc. [4]), are based on distances between the samples in the original vector space. Thus they are faced with the "curse of dimensionality" and the associated sparsity issues, when dealing with very high dimensional data. Recently, some innovative approaches that directly address high-dimensional data mining have emerged. ROCK (Robust Clustering using linKs) [5] 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 which is used to merge clusters. CHAMELEON [6] starts with partitioning the data into a large number of clusters by partitioning the v-nearest neighbor graph. In the subsequent stage clusters are merged based on inter-connectivity and their relative closeness. **Scalability Studies** on clustering have taken two directions:

1. Perform k-means or a variant thereof, on a single computer with limited main memory, with as few scans of the database as possible [7]. These algorithms implicitly assume hyperspherical clusters of about the same size, and thus the key idea is to update sufficient statistics (number of points, sum, sum-squared) about the potential clusters in main memory as one scans the database, and then do further refinement of cluster centers within main memory.
2. Parallel implementations. k-means is readily parallelizeable through data partitioning on distributed memory multicomputers, with little overhead [8]. At each iteration, the current locations of the k means is broadcast to all processors, who then independently perform the time consuming operation of finding the closest mean for each (local) data point, and finally send the (local) updates to the mean positions to a central processor that does a global update using a MPI_allreduce operation.

3 Domain Specific Features and Similarity Space

Notation. Let n be the number of objects (customers) in the data and d the number of features (products) for each sample $\mathbf{x}_j$ with $j \in \{1, \ldots, n\}$. The input data can be represented by a $d \times n$ product-customer matrix $\mathbf{X}$ with the j-th column representing the sample $\mathbf{x}_j$. Hard clustering assigns a label λ_j to each d–dimensional sample $\mathbf{x}_j$, such that similar samples tend to get the same label. The number of distinct labels is k, the desired number of clusters. In general the labels are treated as nominals with no inherent order, though in some cases, such as self-organizing feature maps (SOFMs) or top-down recursive graph-bisection, the labeling may contain extra ordering information. Let $\mathcal{C}_\ell$ denote the set of all customers in the ℓ-th cluster ($\ell \in \{1, \ldots, k\}$), with $\mathbf{x}_j \in \mathcal{C}_\ell \Leftrightarrow \lambda_j = \ell$ and $n_\ell = |\mathcal{C}_\ell|$.

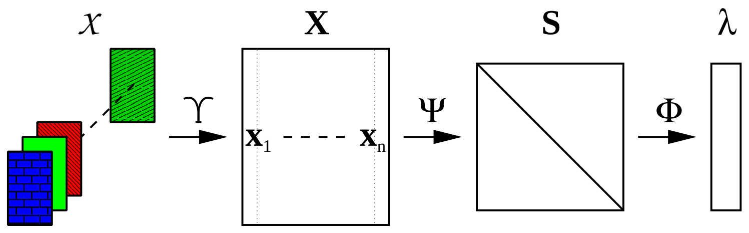


Figure 1. Overview of the similarity based clustering framework OPOSSUM.

Fig. 1 gives an overview of our batch clustering process from a set of raw object descriptions $\mathcal{X}$ via the vector space description $\mathbf{X}$ and similarity space description $\mathbf{S}$ to the cluster labels λ: $(\mathcal{X} \in \mathcal{I}^n) \xrightarrow{\Upsilon} (\mathbf{X} \in \mathcal{F}^n \subset \mathbb{R}^{d \times n}) \xrightarrow{\Psi} (\mathbf{S} \in \mathcal{S}^{n \times n} = [0,1]^{n \times n} \subset \mathbb{R}^{n \times n}) \xrightarrow{\Phi} (\lambda \in \mathcal{O}^n = \{1, \ldots, k\}^n)$.

Feature Selection and Similarity Measures. While most of the well-known clustering techniques [3] have been for numerical features, certain recent approaches assume categorical data [5]. In general, non-binary features are more informative since they capture noisy behavior better for a small number of samples. For example, in market-basket data analysis, a feature typically represents the absence (0) or presence (1) of a particular product in the current basket. However, this treats a buyer of a single corn-flakes box the same as one who buys one hundred such boxes. In OPOSSUM, we extend the common Boolean notation to *non-negative, real-valued features*. The feature $x_{i,j}$ now represents the *volume* of product p_i in a given basket (or sample) $\mathbf{x}_j$. While "volume" could be measured by product quantity, we prefer to use *monetary value* (the product of price and quantity) to quantify feature volume. This yields an almost continuous distribution of feature values for large data sets. More importantly, monetary value represents a normalization across all feature dimensions. This normalization is highly desirable because it better aligns relevance for clustering with retail management objectives.

The key idea behind dealing with very high-dimensional data is to work in similarity space rather than the original vector space in which the feature vectors reside. A similarity measures $\in [0,1]$ 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 $s(\mathbf{x}_a, \mathbf{x}_a) = 1$.

A brute force implementation does involve $O(n^2 \times d)$ operations, since similarity needs to be computed between each pair of data points, and involve all the dimensions. Also, unless similarity is computed on the fly, $O(n^2)$ storage is required for the similarity matrix. However, once this matrix is computed, the following clustering routine does not depend on d at all!

An obvious way to compute similarity is through a suitable monotonic and inverse function of a Minkowski distance, d. Candidates include $s = e^{-d^2}$, and $s(\mathbf{x}_a, \mathbf{x}_b) = 1/(1 + \|\mathbf{x}_a - \mathbf{x}_b\|_2)$. Similarity can also be defined by the angle or cosine of the angle between two vectors. The cosine measure is widely used in text clustering because two documents with equal classification because two documents with equal word composition but different lengths can be considered identical. In retail data this assumption loses important information about the life-time customer value by normalizing them all to 1.

OPOSSUM is based on **Jaccard Similarity**. For binary features, the Jaccard coefficient [2] measures the ratio of the intersection of the product sets to the union of the product sets corresponding to transactions $\mathbf{x}_a$ and $\mathbf{x}_b$:

$$s^{(\mathrm{J})}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^{\dagger}\mathbf{x}_b}{\|\mathbf{x}_a\|_2^2 + \|\mathbf{x}_b\|_2^2 - \mathbf{x}_a^{\dagger}\mathbf{x}_b} \tag{1}$$

Since we want to analyze positive, real-valued features instead, an *extended Jaccard coefficient*, also given by equation 1, but using positive numbers for attribute values, is proposed. This coefficient captures a length-dependent 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 [9], where it is shown that the extended Jaccard coefficient enables us to discriminate by the total value of market-baskets *as well as* to overcome the issues of Euclidean distances in high-dimensional sparse data.

4 CLUSION: Cluster Visualization

Since it is difficult to measure or visualize the quality of clustering in very high-dimensional spaces, we first built a CLUSter visualizatION toolkit, CLUSION, which is briefly described in this section. CLUSION first rearranges the columns and rows of the similarity matrix such that points with the same cluster label are contiguous. It then displays this permuted similarity matrix $\mathbf{S}$ with entries $s_{a,b} = s(\mathbf{x}_a, \mathbf{x}_b)$ as a gray-level image where a black (white) pixel corresponds to minimal (maximal) similarity of 0 (1). The intensity (gray level value) of the pixel at row a and column b corresponds to the similarity between the samples $\mathbf{x}_a$ and $\mathbf{x}_b$. The 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. A bright off-diagonal region may suggest that the clusters in the corresponding rows and columns should be merged. In order to make these regions apparent, thin horizontal and vertical lines are used to show the divisions into the rectangular regions. Visualizing similarity space in this way can help to quickly get a feel for the clusters in the data. Even for a large number of points, a sense for the intrinsic number of clusters k in a data-set can be gained. Examples for CLUSION are given in Fig. 2. For further details, demos, and case studies that support certain design choices (using monetary value, extended Jaccard coefficient), see [1].

5 OPOSSUM

OPOSSUM (Optimal Partitioning of Sparse Similarities Using Metis) is based on partitioning a graph obtained from the similarity matrix, with certain constraints tailored to market-basket data. In particular, it is desirable here to have clusters of roughly equal importance for upper management analysis. Therefore OPOSSUM strives to deliver approximately equal sized (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 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. In this case a cluster represents a k-th fraction of the total feature value $\sum_{j=1}^{n} \sum_{i=1}^{d} x_{i,j}$. If we use extended revenue per product (quantity $\times$ price) as value, then each cluster represents a roughly equal contribution to total revenue.

We formulate the desired balancing properties by assigning each sample (customer) 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 $w_j = 1$. To obtain value balancing properties, a sample $\mathbf{x}_j$'s weight is set to $w_j = \sum_{i=1}^{d} x_{i,j}$. The desired balancing properties have many application driven advantages. However, since natural clusters may not be equal sized, over-clustering (using a larger k) and subsequent merging may be helpful.

5.1 Single-constraint Single-objective Weighted Graph Partitioning

We map the problem of clustering to partitioning a weighted graph with a minimum number of edge cuts while maintaining a balancing constraint. Graphs are a well-understood abstraction and a large body of work exists on partitioning them. In [9] they have been shown to perform superior in high-dimensional document clustering. The objects to be clustered can be viewed as a set of vertices

$\mathcal{V}$. Two web-pages $\mathbf{x}_a$ and $\mathbf{x}_b$ (or vertices v_a and v_b) are connected with an undirected edge $(a, b) \in \mathcal{E}$ of positive weight $s(\mathbf{x}_a, \mathbf{x}_b)$. The cardinality of the set of edges $|\mathcal{E}|$ equals the number of *non-zero* similarities between all pairs of samples. A set of edges whose removal partitions a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ into k pair-wise disjoint sub-graphs $\mathcal{G}_\ell = (\mathcal{V}_\ell, \mathcal{E}_\ell)$, is called an edge separator $\Delta\mathcal{E}$. Our objective is to find such a separator with a minimum sum of edge weights, as given by equation 2.

$$\min_{\Delta\mathcal{E}} \textstyle\sum_{(a,b)\in\Delta\mathcal{E}} s(\mathbf{x}_a, \mathbf{x}_b)\ ,\ \Delta\mathcal{E} = (\mathcal{E} \setminus (\mathcal{E}_1 \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_k)) \tag{2}$$

Without loss of generality, we can assume that the vertex weights w_j are normalized to sum up to 1: $\sum_{j=1}^{n} w_j = 1$ While striving for the minimum cut objective, the constraint $k \cdot \max_\ell \sum_{\lambda_j=\ell} w_j \leq t$ has to be fulfilled. The left hand side quantifies the load balance of the partitioning λ. The load balance is dominated by the worst cluster. A value of 1 indicates perfect balance. Of course, in many cases the constraint can not be fulfilled exactly (e.g., sample balanced partitioning with n odd and k even). However they can be fulfilled within a certain narrow tolerance. We chose the maximum tolerated load imbalance $t \geq 1$ to be 1.05, or 5%, for experiments in section 7.

Finding an optimal partitioning is an NP-hard problem. However, there are very fast, heuristic algorithms for this widely studied problem [10]. The basic approach to dealing with graph partitioning or minimum-cut problems 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 v over to another partition. The decision to move v can be made in time proportional to its degree by simply counting whether more of v's neighbors are on the same partition as v or not. Of course, the desirable side for v will change if many of its neighbors switch, so multiple passes are likely to be needed before the process converges to a local optimum.

After experimentation with several techniques, we decided to use the Metis multi-level multi-constraint graph partitioning package because it is very fast and scales well. A detailed description of the algorithms and heuristics used in Metis can be found in Karypis et al. [11].

5.2 Optimal Clustering

This subsection describes how we find a desirable clustering, with *high overall cluster quality Γ* and a *small number of clusters k*. Our objective is to maximize intra-cluster similarity and minimize inter-cluster similarity, given by
$\text{intra}(\mathbf{X}, \lambda, i) = \frac{1}{(n_i-1)\cdot n_i} \sum_{\lambda_a=\lambda_b=i, a>b} s(\mathbf{x}_a, \mathbf{x}_b)$ and
$\text{inter}(\mathbf{X}, \lambda, i, j) = \frac{1}{n_i \cdot n_j} \sum_{\lambda_a=i, \lambda_b=j} s(\mathbf{x}_a, \mathbf{x}_b)$,
respectively, where i and j are cluster indices. We define our *quality* measure $\Gamma \in [0, 1]$ ($\Gamma < 0$ in case of pathological/inverse clustering) as follows:

$$\Gamma(\mathbf{X}, \lambda) = 1 - \frac{(n-k) \cdot \sum_{i=1}^{k} \sum_{j=i+1}^{k} n_i \cdot \text{inter}(\mathbf{X}, \lambda, i, j)}{n \cdot \sum_{i=1}^{k} (n_i - 1) \cdot \text{intra}(\mathbf{X}, \lambda, i)} \tag{3}$$

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

Finding the "right" number of clusters k for a data set is a difficult, and often ill-posed, problem. 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. To achieve a high quality Γ as well as a low k, the target function $\Lambda \in [0,1]$ is the product of the quality Γ and a penalty term which works very well in practice. Let $n \geq 4$ 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 performance Λ and is defined as $\Lambda(k) = \left(1 - \frac{2k}{n}\right) \cdot \Gamma(k)$. A modest linear penalty was chosen, since our quality criterion does not necessarily improve with increasing k (as compared to 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 performance while increasing k is sufficient.

6 Scalability and Parallel Implementation Issues

The graph metaphor for clustering is not only powerful but can also be implemented in a highly scalable and parallel fashion. In the canonical implementation of Opossum, the most expensive step (in terms of both time and space) is the computation of the similarity measure matrix, rather than the graph-based clustering or post-processing steps! In the straightforward implementation, every pair of samples need to be compared. Consequently, computational time complexity is on the order of $O(n^2 \cdot d)$. In practice, sparsity enables a better (but still $O(n^2)$) performance characteristic. Moreover, if space (memory) is a problem, the similarity matrix can be computed on the fly as the subsequent processing does not involve batch operations.

A given coarse clustering (with a smaller number of clusters than the final k) enables us to limit the similarity computation by only considering object pairs within the same coarse cluster. In retail data such clusterings are often already in place or can be induced fairly easily. Some examples include a pre-segmentation of the customers into geographical regions, a demographic segmentation or an a priori grouping by total revenue. Then, the complexity is reduced from $O(n^2)$ to $O(\Sigma_i n_i^2)$; where n_i are the sizes of the coarse clusters. In particular, if k' coarse clusterings of comparable sizes are present, then computation is reduced by a factor of k'. If such a coarse clustering is not given a priori, a clustering could be computed on a small representative subset of the data. Further incoming data points are then pre-clustered by assigning it to the closest neighboring centroid using the extended Jaccard similarity semantic.

The graph partitioning algorithm for clustering is implemented using Metis. The complexity is essentially determined by the number of edges (customer-pairs with sufficient similarity), and thus it scales linearly with number of customers if the number of non-zero edges per customer is about constant. Note that while Euclidean based similarity induces a fully connected graph, non-Euclidean similarity measures induce several orders of magnitude fewer edges. Two approaches to reduce edges further have been prototyped successfully. On the one hand, edges that do not exceed a domain specific global minimum weight are removed. On the other hand, (if samples are approximately equally important) the v-nearest neighbor subgraph can be created by removing all but the strongest v edges for each vertex. Clearly, this reduces the number of vertices to the order of $O(kn)$. Extensive simulation results comparing Metis with a host of graph partitioning algorithms are given in [11].

Parallel Implementation Considerations. Parallel implementation of the all-pair similarity computation on SIMD or distributed memory processors is trivial. It can be done in a systolic (at step i, compare sample x with sample $x + i \; mod \, n$) 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 in [12], which reports partitioning of graphs with over 7 million vertices (customers) in 7 seconds into 128 clusters on a 128 processor Cray T3E. This shows clearly that our graph-based approach to clustering can be scaled to most real life customer segmentation applications.

7 Experimental Results

We experimented with real retail transactions of 21672 customers of a drugstore. 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` which was removed because it was too common. 473 of these products accounted for less than $25 each in toto and were dropped. The remaining $d = 762$ features and $n = 2466$ customers (34 customers had empty baskets after removing the irrelevant products) were clustered using OPOSSUM.

OPOSSUM's results for this example are obtained with a 550 MHz Pentium II PC with 512 MB RAM in under 10 seconds when similarity is precomputed. Fig. 2 shows the similarity matrix (75% sparse) visualization before (a), after generalized k-means clustering using the standard Jaccard (b), after sample balanced (c), and after value balanced clustering (d). As the relationship-based CLUSION shows, OPOSSUM (c), (d) gives more compact (better separation of on- and off-diagonal regions) and perfectly balanced clusters as compared to, for example, k-means (b). In k-means, the standard clustering algorithm (which can

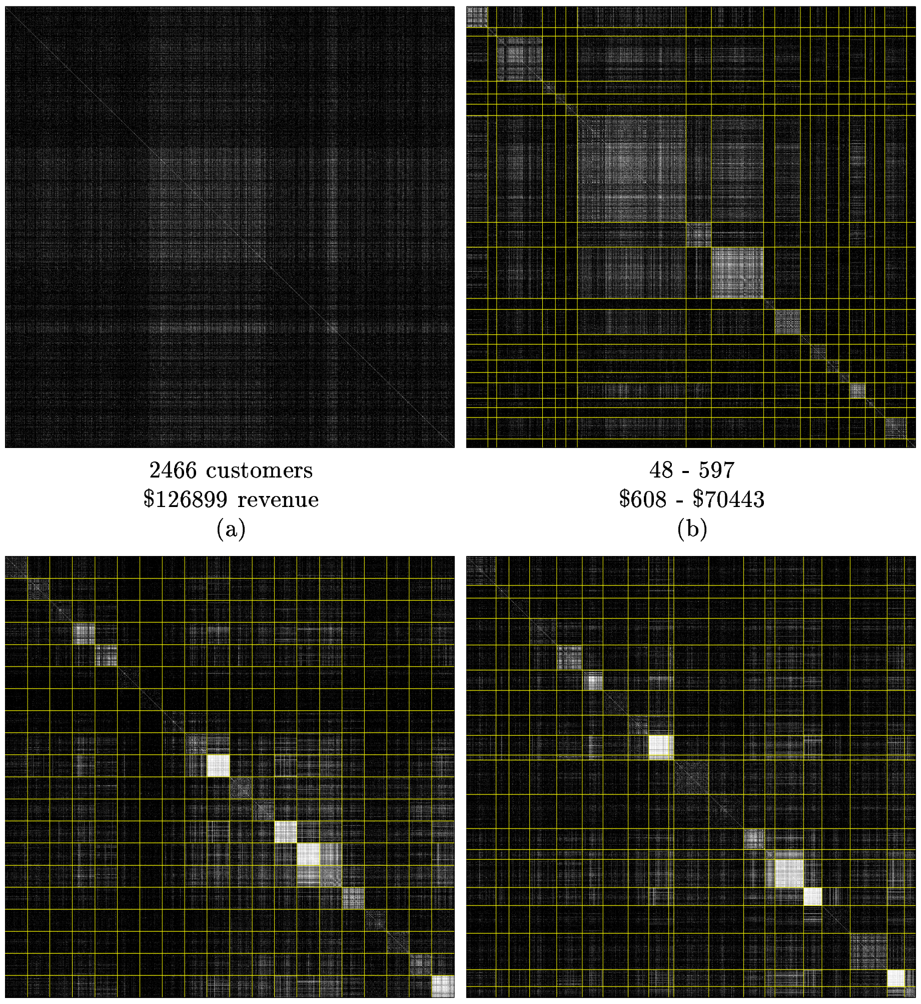
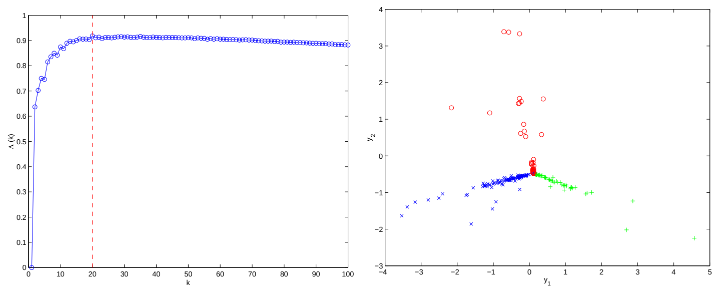


Figure 2. Results of clustering drugstore data. Relationship visualizations using Clusion (a) before, (b) after k-means binary Jaccard, (c) after sample balanced Opossum (d) value balanced Opossum clustering with $k = 20$. In (b) clusters are neither compact nor balanced. In (c) and (d) clusters are very compact as well as balanced.

be generalized by using $-log(s^{(\mathrm{J})})$ as distances), the clusters contain between 48 and 597 customers contributing between \$608 and \$70443 to revenue encumbering a good overview of the customer behavior by marketing. Moreover clusters are hardly compact: Brightness is only slightly better in the on-diagonal regions in (b). All visualizations have been histogram equalized for printing purposes. Fig. 3(a) shows how clustering performance behaves with increasing k. Optimal



(a) (b)

Figure 3. Drugstore data. (a): Behavior of performance Λ for various k using value balanced clustering. The optimal k is found at 20 and is marked with a dashed vertical line. (b): 2–dimensional projection of 762–dimensional data-points on the plane defined by the centroids of the three value-balanced clusters 2 ($\circ$), 9 ($\times$) and 20 ($+$).

performance is found at $k = 20$ for value balanced clustering. In figure 3(b) the data points of the three clusters 2, 9 and 20 is projected onto a 2–dimensional plane defined by the centroids of these three clusters (a la CViz [13]). In this extremely low dimensional projection, the three selected clusters can be reasonably separated using just a linear discriminant function.

Table 1 gives profiles for two of the 20 value balanced customer clusters obtained. A very compact and useful way of profiling a cluster is to look at their most *descriptive* and their most *discriminative* features. This is done by looking at a cluster's highest revenue products and their most unusual revenue drivers. Revenue lift is the ratio of the average spending on a product in a particular cluster to the average spending in the entire data-set. In table 2 the top three descriptive and discriminative products for the customers in all clusters are shown. OPOSSUM identifies customer groups with very similar buying behavior. Marketing can use the customer groups to design and deploy personalized promotional strategies for each group. The clusters are balanced by revenue value and hence provide insight into the contributions and profiles of each customer group.

A detailed comparative study with other clustering methods has been omitted here for lack of space. However, an extensive comparison of several methods on high-dimensional data with similar characteristics, performed recently by us, can be found in [9]. It is very clear that L_p distance based or density estimation based clustering techniques that are representative of the vast majority of data

Table 1. Most *descriptive* (left) and most *discriminative* (right) products purchased by the value balanced clusters $\mathcal{C}_2$ and $\mathcal{C}_9$. Customers in $\mathcal{C}_2$ spent \$10 on average on smoking cessation gum and spent more than 34 times more money on peanuts than the average customer. Cluster $\mathcal{C}_9$ seems to contain strong christmas shoppers probably families with kids.

		value	lift		value	lift
$\mathcal{C}_2$	SMOKING-CESSATION-GUM	10.153	34.732	SMOKING-CESSATION-GUM	10.153	34.732
	TP-CANNING	2.036	18.738	BLOOD-PRESSURE-KITS	1.690	34.732
	BLOOD-PRESSURE-KITS	1.690	34.732	SNACKS/PNTS-NUTS	0.443	34.732
	TP-CASSETTE-RECORDER/PLAY	1.689	9.752	TP-RAZOR-ACCESSORIES	0.338	28.752
	DIABETIC-TESTS	1.521	13.158	BABY-FORMULA-RTF	0.309	28.404
	TP-TOASTER/OVEN/FRY/POP	1.169	7.016	TP-CANNING	2.036	18.738
	BATT-ALKALINE	1.028	1.709	CONSTRUCTION-TOYS	0.855	18.230
	TP-SEASONAL-BOOTS	0.991	1.424	PICNIC	0.379	18.208
	SHELF-CHOCOLATES	0.927	7.924	CG-ETHNIC-FACE	0.350	17.335
	CHRISTMAS-FOOD	0.926	1.988	TP-PLACEMT,NAPKN,CHR-PADS	0.844	16.743

		value	lift		value	lift
$\mathcal{C}_9$	CHRISTMAS-GIFTWARE	12.506	12.986	TP-FURNITURE	0.454	22.418
	CHRISTMAS-HOME-DECORATION	1.243	3.923	TP-ART&CRAFT-ALL-STORES	0.191	13.772
	CHRISTMAS-FOOD	0.965	2.071	TP-FAMILY-PLAN,CONTRACEPT	0.154	13.762
	CHRISTMAS-LIGHT-SETS	0.889	1.340	TP-WOMENS-CANVAS/ATH-SHOE	0.154	13.622
	BOY-TOYS	0.742	1.779	CHRISTMAS-GIFTWARE	12.506	12.986
	AMERICAN-GREETINGS-CARDS	0.702	0.848	TP-CAMERAS	0.154	11.803
	GAMES	0.694	1.639	COMEDY	0.154	10.455
	CHRISTMAS-CANDY	0.680	2.585	CHRISTMAS-CANDOLIERS	0.192	9.475
	TP-SEASONAL-BOOTS	0.617	0.887	TP-INFANT-FORMULA/FOOD	0.107	8.761
	CHRISTMAS-CANDLES	0.601	4.425	CHRISTMAS-MUSIC	0.091	8.625

mining approaches, do not work in the very high-dimensional spaces generated by real-life market-basket data, and that graph partitioning approaches have several advantages in this domain.

8 Concluding Remarks

Opossum efficiently delivers clusters that are balanced in terms of either samples (customers) or value (revenue). Balancing clusters is very useful since each cluster represents a number of data points of similar importance to the user. The associated visualization toolkit Clusion allows managers and marketers to get

Table 2. Overview over *descriptive* (left) and *discriminative* products (right) dominant in each of the 20 value balanced clusters.

1	bath gift packs	hair growth m	boutique island	action items	tp video comedy	family items
2	smoking cessati	tp canning item	blood pressure	smoking cessati	blood pressure	snacks/pnts nut
3	vitamins other	tp coffee maker	underpads hea	underpads hea	miscellaneous k	tp irons items
4	games items	facial moisturi	tp wine jug ite	acrylics/gels/w	tp exercise ite	dental applianc
5	batt alkaline i	appliances item	appliances appl	appliances item	housewares peg	tp tarps items
6	christmas light	appliances hair	tp toaster/oven	multiples packs	christmas light	tv's items
7	christmas food	christmas cards	cold bronchial	sleep aids item	kava kava items	tp beer super p
8	girl toys/dolls	boy toys items	everyday girls	batt rechargeab	tp razors items	tp metal cookwa
9	christmas giftw	christmas home	christmas food	tp furniture it	tp art&craft al	tp family plan
10	christmas giftw	christmas light	pers cd player	pers cd player	tp plumbing ite	umbrellas adult
11	tp laundry soap	facial cleanser	hand&body thera	cat litter scoo	child acetamino	pro treatment i
12	film cameras it	planners/calend	antacid h2 bloc	heaters items	laverdiere ca	ginseng items
13	tools/accessori	binders items	drawing supplie	mop/broom lint	halloween cards	tools/accessori
14	american greeti	paperback items	fragrances op	dental repair k	tp lawn seed it	tp telephones/a
15	american greeti	christmas cards	basket candy it	gift boxes item	hearing aid bat	american greeti
16	tp seasonal boo	american greeti	valentine box c	economy diapers	tp seasonal boo	girls socks ite
17	vitamins e item	group stationer	tp seasonal boo	tp wine 1.5l va	group stationer	stereos items
18	halloween bag c	basket candy it	cold cold items	tp med oint/liq	tp dinnerware i	tp bath towels
19	hair clr perman	american greeti	revlon cls face	hair clr perman	covergirl imple	tp power tools
20	revlon cls face	hair clr perman	headache ibupro	revlon cls face	telephones cord	ardell lashes i

an intuitive visual impression of the group relationships and customer behavior extracted from the data. This is very important for the tool to be accepted and applied by a wider community. The OPOSSUM / CLUSION combine has been successfully applied to several real-life market-baskets. We are currently working on an on-line version of OPOSSUM that incrementally updates clusters as new data points become available. Moreover, modifications for improved scale-up to very large numbers of transactions, using parallel data-flow approaches are currently being investigated.

Acknowledgments. We want to express our gratitude to Mark Davis, Net Perceptions, (`http://www.netperceptions.com`) for providing the drugstore retail data set. This research was supported in part by the NSF under Grant ECS-9000353, and by KD1 and Dell.

References

1. Strehl, A., Ghosh, J.: Value-based customer grouping from large retail data-sets. Proc. SPIE Vol. 4057, (2000) 33–42.
2. Jain, A. K., Dubes, R. C.: Algorithms for Clustering Data. Prentice Hall, New Jersey (1988)
3. Hartigan, J.A.: Clustering Algorithms. Wiley, New York (1975)
4. Rastogi, R., Shim, K.: Scalable algorithms for mining large databases. In Jiawei Han, (ed), KDD-99 Tutorial Notes. ACM (1999)
5. Guha, S., Rastogi, R., Shim, K.: Rock: a robust clustering algorithm for categorical attributes. Proc.15th Int'l Conf. on Data Engineering (1999)
6. Karypis, G., Han, E., Kumar, V.: Chameleon: Hierarchical clustering using dynamic modeling. IEEE Computer, 32(8), (1999) 68–75
7. Bradley, P., Fayyad, U., Reina, C.: Scaling clustering to large databases. In Proc. KDD-98, AAAI Press (1998) 9–15 1998.
8. Dhillon, I., Modha, D.: A data clustering algorithm on distributed memory multiprocessors. KDD Workshop on Large-Scale Parallel Systems (1999)
9. Strehl, A., Ghosh, J., Mooney, R.: Impact of similarity measures on web-page clustering. In Proc. AAAI Workshop on AI for Web Search (2000) 58-64
10. Miller, G.L., Teng, S., Vavasis, S.A., A unified geometric approach to graph separators. In Proc. 31st Annual Symposium on Foundations of Computer Science (1991) 538–547
11. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal of Scientific Computing, 20 **1** (1998), 359–392
12. Schloegel, K., Karypis, G., Kumar, V.: Parallel multilevel algorithms for multi-constraint graph partitioning. Technical Report 99-031, Dept of Computer Sc. and Eng, Univ. of Minnesota (1999)
13. Dhillon, I., Modha, D., Spangler, W.: Visualizing class structure of multidimensional data. In S. Weisberg, editor, Proc. 30th Symposium on the Interface: Computing Science and Statistics, (1998)