Unsupervised Sparse Matrix Co-clustering for Marketing and Sales Intelligence

Anastasios Zouzias[1], Michail Vlachos[2], and Nikolaos M. Freris[2]

[1] Department of Computer Science,
University of Toronto, Canada
[2] IBM Zürich Research Laboratory, Switzerland

Abstract. Business intelligence focuses on the discovery of useful retail patterns by combining both historical and prognostic data. Ultimate goal is the orchestration of more targeted sales and marketing efforts. A frequent analytic task includes the discovery of associations between customers and products. Matrix co-clustering techniques represent a common abstraction for solving this problem. We identify shortcomings of previous approaches, such as the explicit input for the number of co-clusters and the common assumption for existence of a block-diagonal matrix form. We address both of these issues and present techniques for automated matrix co-clustering. We formulate the problem as a recursive bisection on Fiedler vectors in conjunction with an eigengap-driven termination criterion. Our technique does not assume perfect block-diagonal matrix structure after reordering. We explore and identify off-diagonal cluster structures by devising a Gaussian-based density estimator. Finally, we show how to explicitly couple co-clustering with product recommendations, using real-world business intelligence data. The final outcome is a robust co-clustering algorithm that can discover in an automatic manner both disjoint and overlapping cluster structures, even in the preserve of noisy observations.

1 Introduction

Graph structures constitute a prevalent representation form for modeling connections between different entities. In particular, analysis of bi-partite graphs is the focus on a wide spectrum of studies that span from social-network to business analytics and decision making. In business intelligence, bi-partite graphs may capture the connection between sets of customers and sets of products. Analysis of such data holds great importance for companies that collect large amounts of customer interaction data in their data warehouses.

One common analytic process for business intelligence is the identification of groups of customers that buy (or do not buy) a subset of products. The availability of such information is advantageous to both sales and marketing teams, as follows: sales people can use these insights for offering more accurate personalized product suggestions to customers by examining what 'similar' customers buy. In a similar manner, identification of buying/not-buying preferences can

P.-N. Tan et al. (Eds.): PAKDD 2012, Part I, LNAI 7301, pp. 591–603, 2012.

assist marketing people to determine groups of customers interested in a set of package products. This can help organize more focused marketing campaigns, hence leading to a more appropriate allocation of the company's marketing funds.

The described problem can be mapped to a co-clustering instance [1,4,11]. A similar task is 'matrix reordering' which discovers a permutation of matrix rows and columns such that the resulting matrix is as 'compact' as possible. We view co-clustering as a matrix reordering with a subsequent clustering step for dense area identification. For this work we provide examples from a particular setting, where the rows represent customers and the columns identify the products that a customer has bought; but our approach is applicable in different settings, too. An example of a matrix reorganization is shown in Figure 1 a) and b). Existence of a 'one' (black dot) signifies that a customer has bought a product, otherwise the value is 'zero' (white dot). It is evident that the reordered matrix view provides strong evidence on the existence of patterns in the data.

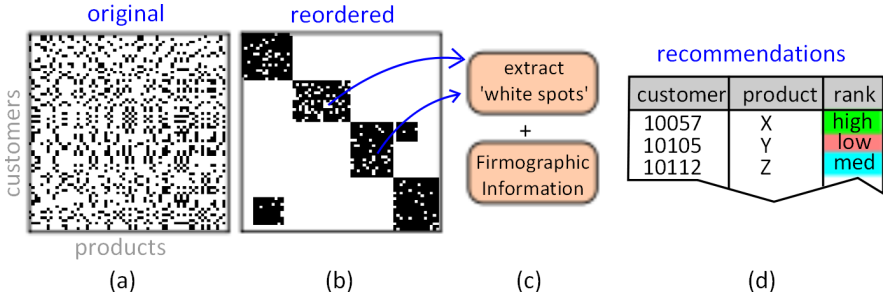


Fig. 1. Overview of our approach. a) Original matrix of customers-products, b) Matrix is reorganized, c) 'White spots' within clusters are extracted and combined with firmographic information, d) Product recommendations are constructed.

Existing co-clustering methods can face several practical issues which limit both their efficiency and the interpretability of the results. For example, the majority of co-clustering algorithms explicitly require as input the number of clusters in the data. In most business scenarios, such an assumption is unrealistic. We cannot assume prior knowledge on the data, but we require a technique that allows data exploration. There exist some methodologies that attempt to perform *automatic* co-clustering [3], i.e., determine the number of co-clusters. They address the problem by evaluating different number of configurations and retaining the solution that provides the best value of the given objective function (e.g., entropy minimization). Such a trial-based approach can significantly affect the performance of the algorithm. Therefore, these techniques are better suited for off-line data processing, rather than for interactive data analysis, which constitutes a key requirement for our setting.

Another shortcoming of many spectral-based approaches is that they typically assume a perfect block-diagonal form for the reordered matrix; "off-diagonal" clusters are usually not detected. This is something that we accommodate in

our solution, where existence of possible "off-diagonal" dense clusters is resolved through a Gaussian-based density estimator algorithm. Because we are interested in finding rectangular clusters, the algorithm discovers the parameters of those rectangles (center, width, height) that best cover the highest density of "off-diagonal" areas. Note, that using such an approach we can also support discovery of overlapping co-clusters with no extra effort.

We use the discovered co-clusters for providing product *recommendations* to customers. The customers in our setting are not individuals but large companies, for whom we have extensive information such as company turnover, number of employees, etc. We use this information to prioritize recommendations. Recall that a co-cluster corresponds to a set of customers with similar buying patterns. To this end, existence of 'white spots' within a discovered co-cluster represents potential product recommendations. However, not all white areas within a cluster are equally important. These recommendations need to be *ranked*. We rank the quality and importance of each recommendation based on:

- The quality of the discovered co-cluster. For example, a single white spot in a very dense and large co-cluster is more important than a white spot in a smaller and sparse co-cluster.
- Firmographic and financial characteristics of the customer; recommendations for customers that have bought more products in the past should be ranked higher, because they have exhibited a higher buying propensity.

With the combination of the above two characteristics, the recommendations exploit both *global* patterns as discovered by the co-clustering, and *personalized* metrics.

In summary, our main **contribution** is providing a robust, unsupervised and fast solution for co-clustering which can be used for interactive business intelligence scenarios. We also demonstrate how to use our solution for providing product recommendations. We perform a comprehensive empirical study using real and synthetic data-sets to validate our solutions. To the best of our knowledge, this is one of the few works that evaluate the performance of co-clustering algorithms on real-world, business intelligence data.

2 Related Work

The principle of co-clustering was introduced first by Hartigan with the goal of 'clustering cases and variables simultaneously' [11]. Initial applications were for the analysis of voting data. Hartigan's method is heuristic in nature and may fail to find existing dense co-clusters under certain cases. In [4] the authors present an iterative algorithm that convergences to a local minimum of the same objective function as in [11]. [1] describes an algorithm which provides constant factor approximations to the optimum co-clustering solution using the same objective function. A spectral co-clustering method based on the Fiedler vector appeared in [6]. Our approach uses a similar analytical toolbox as [6], but in addition is *automatic* (number of clusters need not be given), and does not assume perfect

block-diagonal form for the matrix. A different approach, views the input matrix as an empirical joint probability distribution of two discrete random variables and poses the co-clustering problem as an optimization problem from an information theoretic perspective [7]. So, the optimal co-clustering maximizes the mutual information between the clustered random variables. A method employing a similar metric as the one of [7] appeared in [20]; the latter approach also returns the co-clusters in a hierarchical format. Finally, [16] provides a parallel implementation of the method of [20] using the map-reduce framework. More detailed reviews on the topic can be found in [14] and [21].

Our approach is equally rigorous with the above approaches; more importantly, it lifts important shortcomings of the spectral-based approaches and in addition focuses on the recommendation aspect of co-clustering, something that previous efforts do not consider.

3 Overview of Our Approach

3.1 Preliminaries

We denote by $\mathbf{I}, \mathbf{0}, \mathbf{1}$ the identity matrix, all-zero, and all-one vector, respectively, and the dimensions will become clear from the context. We define $[m] := \{1, 2, \ldots, m\}$. Let $\boldsymbol{C}$ be an $m \times n$ matrix. For a subset of its rows R and a subset of its columns T we denote by $\boldsymbol{C}_{R,T}$ the sub-matrix formed by rows R and columns T.

Given an undirected graph $G = (V, E)$ on n vertices with adjacency matrix $\boldsymbol{A}$, its Laplacian matrix is defined $\boldsymbol{L} := \boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D}$ is a diagonal matrix of size n with $D_{ii} = \sum_j A_{ij}$. Moreover, the normalized Laplacian matrix of G is defined, when $D_{ii} > 0$ for all i, as $\widehat{\boldsymbol{L}} := \mathbf{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$. The eigenvector of $\widehat{\boldsymbol{L}}$ that corresponds to the second smallest eigenvalue of $\widehat{\boldsymbol{L}}$ is known as the *Fiedler vector* [8]. Let $(S, \bar{S})$ be a bipartition of the vertex set of G. Denote by $\mathrm{cut}(S, \bar{S})$ the sum of weights of the edges between the sets S and $\bar{S}$.

3.2 Graph Partitioning

Partitioning a graph into two balanced vertex sets (i.e., two sets such that neither is much larger than the other one) while minimizing the number of edges between them is a fundamental combinatorial problem with various applications [19]. Choosing a particular *balancing condition* gives rise to different related measures of the quality of the cut including conductance, expansion, normalized and sparsest cut. The two most commonly used balancing objective functions are the *ratio cut* [10] and the *normalized cut* [17]. In ratio cut, the size of a subset $S \subset V$ of a graph is measured by its number of vertices $|S|$, where in normalized cut the size is measured by the total weights of its edges, denoted by $\mathrm{vol}(S)$. Here we will use the normalized cut objective function

$$\mathrm{Ncut}(S, \bar{S}) := \frac{\mathrm{cut}(S, \bar{S})}{\mathrm{vol}(S)} + \frac{\mathrm{cut}(S, \bar{S})}{\mathrm{vol}(\bar{S})}.$$

Note that the above objective function typically takes a small value if the bi-partition $(S, \bar{S})$ is not balanced, hence it favors balanced partitions. The goal of graph partitioning is to solve the optimization problem

$$\min_{S \subset V} \mathrm{NCut}(S, \bar{S}). \tag{1}$$

This problem is NP-hard. Many approximation algorithms for this problem have been developed over the last years [2,12], however they turn out not to be performing well in practice. In our approach, we employ a heuristic that is based on spectral techniques and works well in practice [13,9]. Following the notation of [13], for any $S \subset V$, define the vector $\boldsymbol{q} \in \mathbb{R}^n$ as

$$q_i = \begin{cases} +\sqrt{\eta_2/\eta_1}, & i \in S; \\ -\sqrt{\eta_1/\eta_2}, & i \in \bar{S}, \end{cases} \tag{2}$$

where $\eta_1 = \mathrm{vol}(S)$ and $\eta_2 = \mathrm{vol}(\bar{S})$. The objective function in (1) can be written (see [6] for details) as follows

$$\min \frac{\boldsymbol{q}^\top \boldsymbol{L}\boldsymbol{q}}{\boldsymbol{q}^\top \boldsymbol{D}\boldsymbol{q}}, \quad \text{subject to } \boldsymbol{q} \text{ as in (2) and } \boldsymbol{q}^\top \boldsymbol{D}\mathbf{1} = \mathbf{0},$$

where the extra constraint $\boldsymbol{q}^\top \boldsymbol{D}\mathbf{1} = \mathbf{0}$ excludes the trivial solution where $S = V$ or $S = \emptyset$. Even though the above problem is also NP-hard, we adopt a spectral 2-clustering heuristic: first, we drop the constraint that $\boldsymbol{q}$ is as in Eqn. (2); this relaxation is equivalent to finding the second largest eigenvalue and eigenvector of the generalized eigensystem $\boldsymbol{L}\boldsymbol{z} = \lambda \boldsymbol{D}\boldsymbol{z}$. Then, we round the resulting eigenvector $\boldsymbol{z}$ (a.k.a. *Fiedler vector*) to obtain a bipartition of G. The rounding is performed by applying 2-means clustering separately on the coordinates of $\boldsymbol{z}$, which can be solved *exactly* and efficiently.

4 The Algorithm

The proposed algorithm consists of two steps: in the first step, we compute a permutation of the row set and column set using a recursive spectral bi-partitioning algorithm; in the second step we use the permuted input matrix to identify any remaining clusters by means of a Gaussian-based density estimator.

4.1 Recursive Spectral Bi-partitioning

In this section, we recall a graph theoretic approach to the co-clustering problem [6]. The input is an $m \times n$ matrix $\boldsymbol{C}$. For illustration purposes, we assume that $\boldsymbol{C}$ is a binary matrix, i.e., its elements are in $\{0, 1\}$, but our approach is applicable in general. Given $\boldsymbol{C}$, we can uniquely define a bipartite graph $G = (L \cup R, E)$, $|L| = m$, $|R| = n$ as follows: Each element of the left set of vertices, L, corresponds to a row of $\boldsymbol{C}$ and each element of the right vertices R to a column of $\boldsymbol{C}$. We connect an edge between $i \in L$ and $j \in R$ if and only if $C_{ij} = 1$. Now given the bipartite graph G corresponding to $\boldsymbol{C}$, we find a balanced cut in G with few edges crossing the cut.

Algorithm 1. Recursive Spectral Bipartition

1: **procedure** RecBipart($\boldsymbol{C}$) ▷ $\boldsymbol{C}$: an $m \times n$ binary matrix, *EigenGap* : a stopping parameter
2: $\{(R_1, R_2), (T_1, T_2)\}$ = SplitCluster($\boldsymbol{C}$).
3: **if** there is a split (R_1, R_2) and (T_1, T_2) **then**
4: Run RecBipart($\boldsymbol{C}_{R_1,T_1}$)
5: Run RecBipart($\boldsymbol{C}_{R_2,T_2}$)
6: **end if**
7: **Output:** A partition of the row and column set of $\boldsymbol{C}$, i.e., $\cup R_i = [m]$ and $\cup T_i = [n]$.
8: **end procedure**

9: **procedure** SplitCluster($\boldsymbol{C}$) ▷ $\boldsymbol{C}$: binary matrix
10: Let $\widehat{\boldsymbol{L}}$ be the normalized Laplacian of the bipartite graph that corresponds to $\boldsymbol{C}$
11: Let λ_2 and $\boldsymbol{z}$ be the second smallest eigenvalue and eigenvector of $\widehat{\boldsymbol{L}}$
12: **if** $\lambda_2 > EigenGap$ **then**
13: Bipartition the coordinates of $\boldsymbol{z}$ s.t. the sum of its intra-variances is minimized.
14: **end if**
15: **Output:** A bipartition of the row set and the column set into (R_1, R_2) and (T_1, T_2), respectively.
16: **end procedure**

4.2 Eigengap-Based Termination

A basic fact in spectral graph theory is that the number of connected components in an undirected graph equals to the multiplicity of the zero eigenvalue of its normalized Laplacian matrix. Cheeger's inequality provides an "approximate" version of the latter fact [5]. That is, a graph has a sparse (normalized) cut if and only if there are at least two eigenvalues that are close to zero. Let the conductance of a graph $G = (V, E)$ defined as follows

$$c(G) := \min_{S \subseteq V, \mathrm{vol}(S) \leq \frac{|E|}{2}} \frac{\mathrm{cut}(S, \bar{S})}{\mathrm{vol}(S)}.$$

Cheeger's inequality [5] tells us that $2c(G) \geq \lambda_2 \geq c(G)^2/2$, where λ_2 is the second smallest eigenvalue of the normalized Laplacian of G. The first inequality of the above equation implies that if λ_2 is large, then G does not have sufficiently small conductance. The latter implication supports our choice of the termination criterion of the recursion of Algorithm 1 (see Step 11 of the SplitCluster procedure). Roughly speaking, we want to stop the recursion when the matrix can not be reduced (after permutation of rows and columns) to an approximately block diagonal matrix, equivalently when the bipartite graph associated to the current matrix does not contain any sparse cut. Using Cheeger's inequality, we can efficiently check if the bipartite graph has a sufficiently good cut or not. An illustration of the algorithm's recursion is explored in Fig.2. Our approach has many similarities with Newman's modularity partitioning but it is not identical [15].

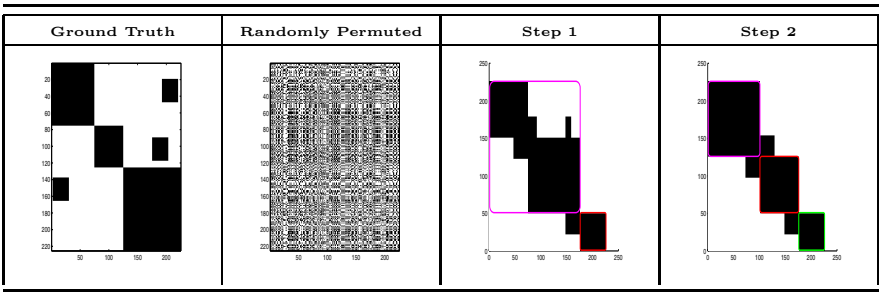


Fig. 2. Running example of Algorithm 1

4.3 Discovering Off-Diagonal Clusters

After termination of Algorithm 1, we expect to have produced a fairly good reordering of the rows and columns of the input matrix $\boldsymbol{C}$ and moreover to have discovered a set of co-clusters that have disjoint row and column sets. However, in almost all instances of practical interest, we cannot expect the set of co-clusters to have disjoint row and column sets; several "off-diagonal" co-clusters may have appeared after the course of our reordering algorithm. In order not to discard this potentially useful information, we apply as a post-processing step a Gaussian-based density estimator on the matrix after removing the set of co-clusters already discovered by Algorithm 1. This process is depicted in Figure 3. In the first step, we remove all the clusters that have been already extracted by Algorithm 1. In the second step we apply a density estimator to discover any (possibly) remaining co-clusters. That is, we convolute a Gaussian mask over all positions of the binary matrix to detect the most dense areas. Initially, we set a sufficiently large size on the Gaussian mask[1]. We then progressively reduce the size of the mask by a fixed constant. At each step, we record all sufficiently dense areas (those for which the convolution exceeds some threshold) and remove the co-cluster from further consideration.

Complexity: We briefly discuss the time complexity of Algorithm 1. For ease of presentation, assume that the input is a square matrix of size n and let $T(n)$ be the time complexity. Moreover, we may assume[2] that in every recursion step the partitioning is balanced. Since the input matrix is sparse, the following recursion holds $T(n) \approx 2T(n/2) + \mathcal{O}(n \log n)$, where the extra additive factor is due to sorting (Lanczos method is used for computing the eigenvector). Solving the recursion we get that $T(n) = \mathcal{O}(n \log^2 n)$ which implies that our method is only more expensive than a single bi-partitioning by a poly-logarithmic factor.

[1] We can over-estimate the size of the largest dense rectangle by the number of non-zero entries of the input matrix.

[2] We are allowed to do so, since a constant number of successive unbalanced cuts indicate that the recursion should terminate.

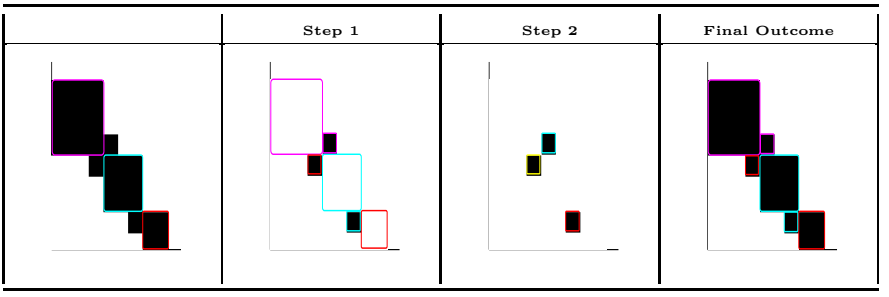


Fig. 3. Discovering Off-diagonal Clusters

4.4 Recommendations

Algorithm 1 together with the density estimation step output a list of co-clusters. In the business scenarios that we consider co-clusters will represent strongly correlated customer and products subsets. We illustrate how this information can be used to drive meaningful product recommendations.

Many discovered co-clusters are expected to contain "white-spots". These represent customers that exhibit similar buying pattern with a number of other customers, they still have not bought a product within the co-cluster. These are products that constitute good recommendations. Essentially, we exploit the existence of *globally-observable* patterns for making individual recommendations.

Not all "white-spots" are equally important. We rank them by considering firmographic and financial characteristics of the customers. The intuition is that 'wealthy' customers/companies that have bought many products in the past are better-suited candidates. They are at financial position to buy a product and they have already established a buying relationship. In our formula we consider three factors:

- **Turnover T** is the revenue of the company as provided in its financial statements.
- **Past Revenue R** is the revenue amount that our company made in its interactions with the customer during the past 3 years.
- **Industry Growth IG** represents that predicted growth for the industry in which the customer belongs for the upcoming year. This data is furnished from marketing databases and is estimated from diverse global financial indicators.

Therefore the rank r of a given white-spot that captures a customer-product recommendation is given by:

$$r = w_1 \mathrm{T} + w_2 \mathrm{R} + w_3 \mathrm{IG}, \quad \sum_i w_i = 1,$$

In our scenario, the weights w_1,w_2,w_3 are assumed to be equal but in general they can be tuned appropriately.

We have described how to rank the "white-spots" within a particular co-cluster. In order to give a total ordering on the set of the recommendations, we should normalize these ranking value with the importance of each co-cluster. We define the importance of each co-cluster as the product of its area and density normalized by the sum of the importance of all co-clusters. Hence, we normalize all recommendations by the importance of the corresponding co-cluster.

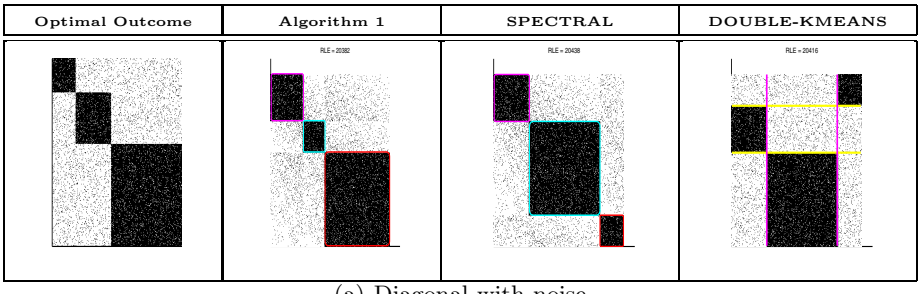
5 Experiments

5.1 Comparison with other Techniques

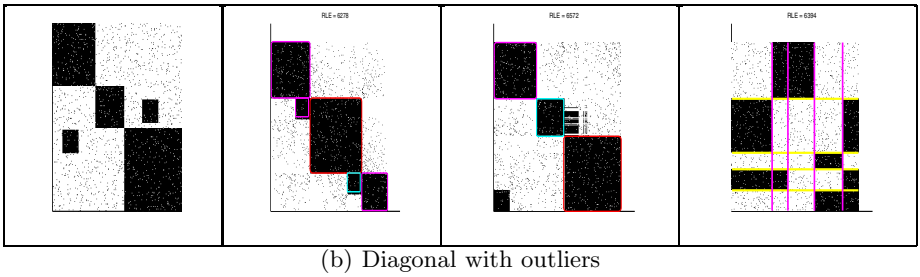
We compare the proposed approach with two other techniques. The first one (SPECTRAL) is described in [6] and is similar with the proposed approach. The main difference is that the approach of [6] performs k-partition of the input matrix using the eigenvectors that correspond to the smallest eigenvalues. Moreover, in order to compute the clustering it utilizes k-means clustering which makes the approach randomized, compared to our approach which is deterministic. The second one (DOUBLE-KMEANS) is described in [1]. First, it performs k-means clustering using as input vectors the columns of the input matrix and then permutes the columns by grouping together columns that belong in the same cluster. In the second step, it performs the same procedure on the rows of the input matrix using a possible different number of clusters, say l. This approach outputs $k \cdot l$ clusters. Typical values for k and l that we use, are between 3 and 5. We run all the above algorithms on synthetic data which we produced by creating several block-diagonal and off-diagonal clusters. We introduced "salt-and-pepper" noise in the produced matrix, in an effort to examine the accuracy of the compared algorithms even in when diluting the strength of the original patterns. The results are summarized in Figure 4. We observe that our algorithm can detect with high efficiency the original patterns, whereas the original spectral and k-Means algorithms present results of lower quality.

5.2 Compression-Based Evaluation

It is common to judge the effectiveness of a particular algorithm based on the value of an objective function. However, it is not clear how to evaluate the effectiveness of various co-clustering algorithms that are designed to optimize different objective functions. It is even harder to fairly compare the quality of two algorithms that output a different number of co-clusters. Therefore, we make a comparison using compressibility metrics: we measure how many bytes the reordered array will require when stored using Run-Length-Encoding (RLE) [18]. Recall that RLE replaces long blocks of repetitive values with just two numbers: the value and the length of the "run". For example, RLE encodes the sequence $0, 0, 0, 0, 0, 0, 0, 0, 0, 0$ as a tuple $(9, 0)$. This simple metric allows to quantify how appropriately each algorithm packed together zeros and ones. Understandably, larger number of bytes for the reordered matrix under RLE compression, indicates worse performance in placing zeros and ones in adjacent positions.



(a) Diagonal with noise



(b) Diagonal with outliers

Fig. 4. The first column contains the ground truth and the remaining columns contains the output of the three algorithms described in Section 5.1. All algorithms take as input a randomly permuted (independently in rows and columns) version of the ground truth instance.

The results using are depicted in Table 1. For this we extract matrices that represent buying patterns within our company for various industries of customers, because different industries exhibit different patterns. We notice that the proposed recursive algorithm results in compressed matrix sizes significantly smaller than the competitive approaches, suggesting a more effective co-clustering process.

Table 1. We compare the efficacy of the various co-clustering algorithms by reporting the number of bytes when the reordered matrix is compressed using Run-Length-Encoding (RLE). We present results for buying patterns extracted from various customer industries. Our approach results in reordered matrices that can be better compressed.

Industry's name	Original	Our Approach	SPECTRAL	DOUBLE-KMEANS
Computer Services	2004	108	306	544
Professional Services	1136	212	484	658
Banks	2810	372	1012	1348
Provincial Government	954	128	236	352
Other Productions Ind.	1458	288	648	720
Retail	5232	360	888	2148
Travel & Transport	1158	204	534	582
Wholesale distribution services	638	212	408	394

5.3 Business Intelligence on Real Datasets

For this example we use real-world data provided by our sales department relating to approximately 30,000 Swiss customers. The dataset contains all firmographic information pertaining to the customers, such as: industry categorization (electronic, automotive, etc), expected industry growth, customer's turnover for last, past revenue. We perform co-clustering on the customer-product matrix.. We apply our algorithm on each industry separately, because sales people only have access to their industry of specialization. Figure 5 shows the outcome of the algorithm and the detected diagonal and off-diagonal clusters. The highest ranked recommendations are detected within the blue cluster, and they suggest that 'white-spot' customers within this cluster can be approached with an offer for the product 'System-I'. These customers were ranked higher based on their financial characteristics.

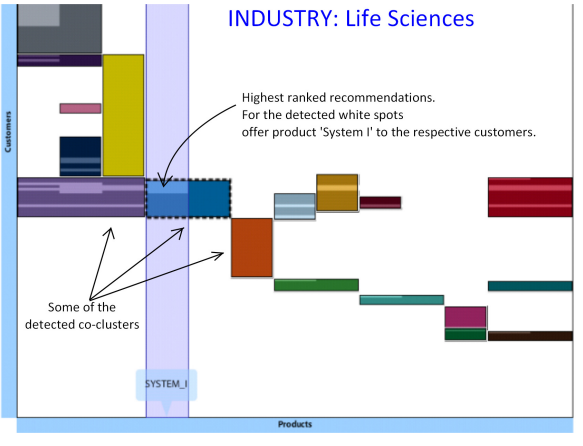


Fig. 5. Example of our co-clustering algorithm when applied on customers of a particular industry (Life Sciences). We see that the algorithm can easily discern off-diagonal clusters. For the illustrated "white-spot" customers within the blue cluster, there is a product recommendation for 'System I'.

6 Conclusion

Focus of this work was to explicitly show how co-clustering techniques can be coupled with recommender systems for business intelligence applications. Contributions of our approach include:

– An unsupervised spectral-based technique for detection of large 'diagonal' co-clusters. We present a robust termination criterion and we depict its accuracy on a variety of synthetic data where we compare with ground-truth.

– A Gaussian-based density estimator for identification of smaller 'off-diagonal' co-clusters.
– A comprehensive comparison of our approach with prevalent co-clustering approaches using a compression-based metric.
– A direct application of our methodology in business recommender systems.

References

1. Anagnostopoulos, A., Dasgupta, A., Kumar, R.: Approximation Algorithms for co-Clustering. In: Proceedings of ACM Symposium on Principles of Database Systems (PODS), pp. 201–210 (2008)
2. Arora, S., Rao, S., Vazirani, U.: Expander Flows, Geometric Embeddings and Graph Partitioning. J. ACM 56, 5:1–5:37 (2009)
3. Chakrabarti, D., Papadimitriou, S., Modha, D.S., Faloutsos, C.: Fully Automatic Cross-associations. In: Proc. of International Conference on Knowledge Discovery and Data Mining (KDD), pp. 79–88 (2004)
4. Cho, H., Dhillon, I.S., Guan, Y., Sra, S.: Minimum Sum-Squared Residue co-Clustering of Gene Expression Data. In: Proc. of SIAM Conference on Data Mining, SDM (2004)
5. Chung, F.R.K.: Spectral Graph Theory. American Mathematical Society (1994)
6. Dhillon, I.S.: Co-Clustering Documents and Words using Bipartite Spectral Graph Partitioning. In: Proc. of International Conference on Knowledge Discovery and Data Mining (KDD), pp. 269–274 (2001)
7. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-Clustering. In: Proc. of International Conference on Knowledge Discovery and Data Mining (KDD), pp. 89–98 (2003)
8. Fiedler, M.: Algebraic Connectivity of Graphs. Czechoslovak Mathematical Journal 23(98), 298–305 (1973)
9. Guattery, S., Miller, G.L.: On the Performance of Spectral Graph Partitioning Methods. In: Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 233–242 (1995)
10. Hagen, L., Kahng, A.: New Spectral Methods for Ratio Cut Partitioning and Clustering. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 11(9), 1074–1085 (1992)
11. Hartigan, J.A.: Direct Clustering of a Data Matrix. Journal of the American Statistical Association 67(337), 123–129 (1972)
12. Leighton, T., Rao, S.: Multicommodity Max-flow Min-cut Theorems and their Use in Designing Approximation Algorithms. J. ACM 46, 787–832 (1999)
13. Luxburg, U.: A Tutorial on Spectral Clustering. Statistics and Computing 17, 395–416 (2007)
14. Madeira, S., Oliveira, A.L.: Biclustering Algorithms for Biological Data Analysis: a survey. Trans. on Comp. Biology and Bioinformatics 1(1), 24–45 (2004)
15. Newman, M.E.J.: Fast Algorithm for Detecting Community Structure in Networks. Phys. Rev. E 69, 066133 (2004)
16. Papadimitriou, S., Sun, J.: DisCo: Distributed Co-clustering with Map-Reduce: A Case Study towards Petabyte-Scale End-to-End Mining. In: Proc. of International Conference on Data Mining (ICDM), pp. 512–521 (2008)

17. Shi, J., Malik, J.: Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
18. Salomon, D.: Data Compression: The Complete Reference, 2nd edn. Springer-Verlag New York, Inc. (2000)
19. Shmoys, D.B.: Cut Problems and their Application to Divide-and-conquer, pp. 192–235. PWS Publishing Co. (1997)
20. Sun, J., Faloutsos, C., Papadimitriou, S., Yu, P.S.: GraphScope: Parameter-free Mining of Large Time-evolving Graphs. In: Proc. of KDD, pp. 687–696 (2007)
21. Tanay, A., Sharan, R., Shamir, R.: Biclustering Algorithms: a survey. Handbook of Computational Molecular Biology (2004)