

Biclustering in data mining

Stanislav Busygin^a, Oleg Prokopyev^{b,*}, Panos M. Pardalos^a

^aDepartment of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611, USA

^bDepartment of Industrial Engineering, University of Pittsburgh, Pittsburgh, PA 15261, USA

Available online 6 February 2007

Abstract

Biclustering consists in simultaneous partitioning of the set of samples and the set of their attributes (features) into subsets (classes). Samples and features classified together are supposed to have a high relevance to each other. In this paper we review the most widely used and successful biclustering techniques and their related applications. This survey is written from a theoretical viewpoint emphasizing mathematical concepts that can be met in existing biclustering techniques.

© 2007 Published by Elsevier Ltd.

Keywords: Data mining; Biclustering; Classification; Clustering; Survey

1. The main concept

Due to recent technological advances in such areas as IT and biomedicine, the researchers face ever-increasing challenges in extracting relevant information from the enormous volumes of available data [1]. The so-called *data avalanche* is created by the fact that there is no concise set of parameters that can fully describe a state of real-world complex systems studied nowadays by biologists, ecologists, sociologists, economists, etc. On the other hand, modern computers and other equipment are able to produce and store virtually unlimited data sets characterizing a complex system, and with the help of available computational power there is a great potential for significant advances in both theoretical and applied research. That is why in recent years there has been a dramatic increase in the interest in sophisticated *data mining* and *machine learning* techniques, utilizing not only statistical methods, but also a wide spectrum of computational methods associated with large-scale optimization, including algebraic methods and neural networks.

The problems of partitioning objects into a number of groups can be met in many areas. For instance, the vector partition problem, which consists in partitioning of n d -dimensional vectors into p parts has broad expressive power and arises in a variety of applications ranging from economics to symbolic computation (see, e.g., [2–4]). However, the most abundant area for the partitioning problems is definitely data mining. Data mining is a broad area covering a variety of methodologies for analyzing and modeling large data sets. Generally speaking, it aims at revealing a genuine similarity in data profiles while discarding the diversity irrelevant to a particular investigated phenomenon. To analyze patterns existing in data, it is often desirable to partition the data samples according to some similarity criteria. This task is called *clustering*. There are many clustering techniques designed for a variety of data types—homogeneous and

* Corresponding author.

E-mail addresses: busygin@ufl.edu (S. Busygin), prokopyev@engr.pitt.edu (O. Prokopyev), pardalos@ufl.edu (P.M. Pardalos).

nonhomogeneous numerical data, categorical data, 0–1 data. Among them one should mention hierarchical clustering [5], k -means [6], self-organizing maps (SOM) [7], support vector machines (SVM) [8,9], logical analysis of data (LAD) [10,11], etc. A recent survey on clustering methods can be found in [12].

However, working with a data set, there is always a possibility to analyze not only properties of samples, but also of their components (usually called *attributes* or *features*). It is natural to expect that each associated part of samples recognized as a cluster is induced by properties of a certain subset of features. With respect to these properties we can form an associated cluster of features and bind it to the cluster of samples. Such a pair is called a *bicluster* and the problem of partitioning a data set into biclusters is called a *biclustering* problem.

In this paper we review the most widely used and successful biclustering techniques and their related applications. Previously, there were published few surveys on biclustering [13,14], as well as a Wikipedia article [15]. However, we tried to write this survey from a more theoretical viewpoint emphasizing mathematical concepts that can be found in existing biclustering techniques. In addition this survey discusses recent developments not included in the previous surveys and includes references to public domain software available for some of the methods and most widely used benchmarks data sets.

2. Formal setup

Let a data set of n samples and m features be given as a rectangular matrix $A = (a_{ij})_{m \times n}$, where the value a_{ij} is the expression of i th feature in j th sample. We consider classification of the samples into classes

$$\begin{aligned} \mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_r, \quad \mathcal{S}_k \subseteq \{1, \dots, n\}, \quad k = 1, \dots, r, \\ \mathcal{S}_1 \cup \mathcal{S}_2 \cup \dots \cup \mathcal{S}_r = \{1, \dots, n\}, \\ \mathcal{S}_k \cap \mathcal{S}_\ell = \emptyset, \quad k, \ell = 1, \dots, r, \quad k \neq \ell. \end{aligned}$$

This classification should be done so that samples from the same class share certain common properties. Correspondingly, a feature i may be assigned to one of the feature classes

$$\begin{aligned} \mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_r, \quad \mathcal{F}_k \subseteq \{1, \dots, m\}, \quad k = 1, \dots, r, \\ \mathcal{F}_1 \cup \mathcal{F}_2 \cup \dots \cup \mathcal{F}_r = \{1, \dots, m\}, \\ \mathcal{F}_k \cap \mathcal{F}_\ell = \emptyset, \quad k, \ell = 1, \dots, r, \quad k \neq \ell, \end{aligned}$$

in such a way that features of the class \mathcal{F}_k are “responsible” for creating the class of samples \mathcal{S}_k . Such a simultaneous classification of samples and features is called *biclustering* (or *co-clustering*).

Definition 1. A *biclustering* of a data set is a collection of pairs of sample and feature subsets $\mathcal{B} = ((\mathcal{S}_1, \mathcal{F}_1), (\mathcal{S}_2, \mathcal{F}_2), \dots, (\mathcal{S}_r, \mathcal{F}_r))$ such that the collection $(\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_r)$ forms a partition of the set of samples, and the collection $(\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_r)$ forms a partition of the set of features. A pair $(\mathcal{S}_k, \mathcal{F}_k)$ will be called a *bicluster*.

It is important to note here that in some of the biclustering methodologies a direct one to one correspondence between classes of samples and classes of features is not required. Moreover, the number of sample and feature classes is allowed to be different. This way we may consider not only pairs $(\mathcal{S}_k, \mathcal{F}_k)$, but also other pairs $(\mathcal{S}_k, \mathcal{F}_\ell)$, $k \neq \ell$. Such pairs will be referred to as *co-clusters*. Another possible generalization is to allow *overlapping* of co-clusters.

The criteria used to relate clusters of samples and clusters of features may have different nature. Most commonly, it is required that the submatrix corresponding to a bicluster either is overexpressed (i.e., mostly includes values above average), or has a lower variance than the whole data set, but in general, biclustering may rely on any kind of common patterns among elements of a bicluster.

3. Visualization of biclustering

One popular tool for visualizing data sets is *heatmaps*. A heatmap is a rectangular grid composed of pixels each of which corresponds to a data value. The color of a pixel ranges between bright green or blue (lowest values) and bright red (highest values) visualizing the corresponding data value. This way, if the samples or/and features of the data set are ordered with respect to some pattern in the data, the pattern becomes obvious to observe visually.

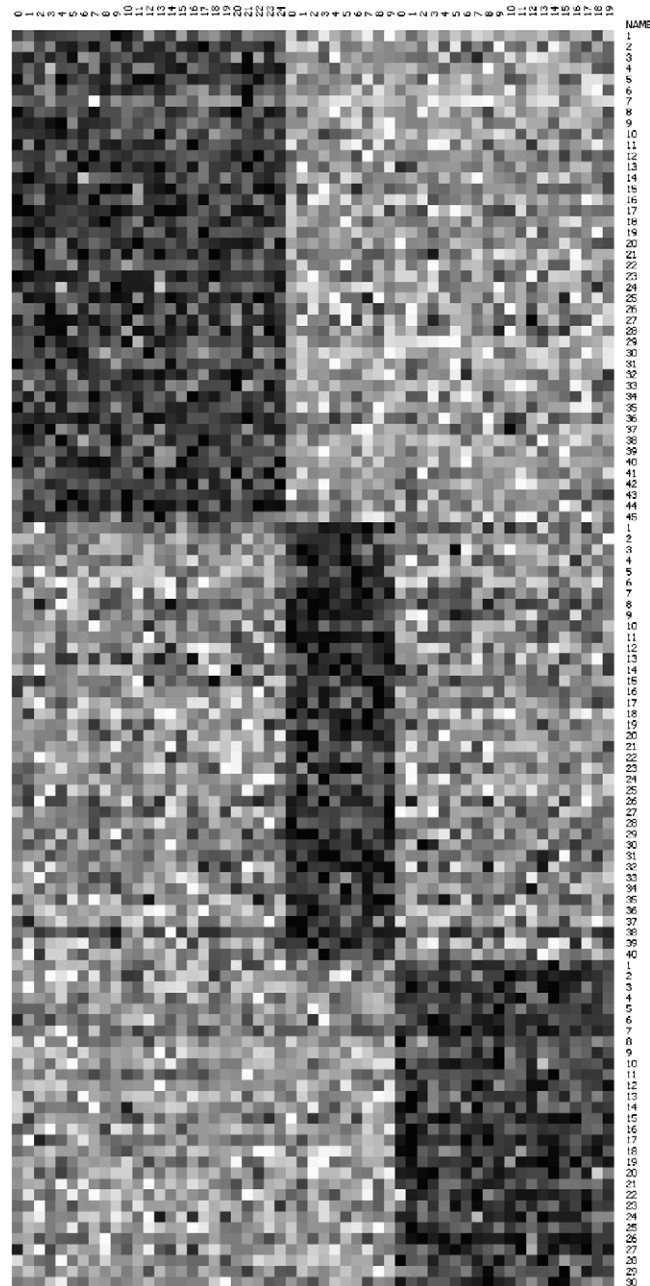


Fig. 1. Partitioning of samples and features into three clusters.

When one constructs a reasonable biclustering of a data set and then reorders samples and features by cluster numbers, the heatmap is supposed to show a “checkerboard” pattern as diagonal blocks show biclusters that are the distinguished submatrices according to the used biclustering method. Fig. 1 is an example of data set with three biclusters of overexpressed values visualized as the heatmap (in the black-and-white diagram darker pixels correspond to higher values).

To create a heatmap, one can use *Heatmap Builder* software [16].

4. Relation to SVD

Singular value decomposition (SVD) is a remarkable matrix factorization which generalizes eigendecomposition of a symmetric matrix providing the orthogonal basis of eigenvectors. SVD is applicable to any rectangular matrix $A = (a_{ij})_{m \times n}$. It delivers orthogonal matrices $U = (u_{ik})_{m \times p}$ and $V = (v_{jk})_{n \times p}$ (i.e., the columns of the matrices are orthogonal to each other and have the unit length) such that

$$U^T A V = \text{diag}(\sigma_1, \dots, \sigma_p), \quad p = \min(m, n). \quad (1)$$

The numbers $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0$ are called *singular values*, the columns of U are called *left singular vectors* and the columns of V are called *right singular vectors* of A . This way, left singular vectors provide an orthonormal basis for columns of A , and right singular vectors provide an orthonormal basis for rows of A . Moreover, these bases are coupled so that

$$A v_k = \sigma_k u_k,$$

$$A^T u_k = \sigma_k v_k,$$

where u_k is k th left singular vector, and v_k is k th right singular vector of the matrix. The singular values of A are precisely the lengths of the semi-axes of the hyperellipsoid $E = \{Ax : \|x\|_2 = 1\}$.

The SVD provides significant information about properties of the matrix. In particular, if σ_r is the last nonzero singular value (i.e., $\sigma_{r+1} = \dots = \sigma_p = 0$), then

$$\text{rank}(A) = r,$$

$$\text{null}(A) = \text{span}\{v_{r+1}, \dots, v_n\},$$

$$\text{ran}(A) = \text{span}\{u_1, \dots, u_r\},$$

where $\text{span}\{x_1, \dots, x_k\}$ denotes the linear subspace spanned by the vectors x_1, \dots, x_k , $\text{null}(A) = \{x : Ax = 0\}$ is the nullspace of the matrix, and $\text{ran}(A)$ is the linear subspace spanned by the columns of A . It is easy to see from these properties that the SVD is a very useful tool for dimensionality reduction in data mining. Taking also into account that the Frobenius norm of the matrix

$$\|A\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n a_{ij}^2 = \sum_{k=1}^r \sigma_k^2,$$

one can obtain the best in sense of Frobenius norm low-rank approximation of the matrix by equating all singular values after some σ_ℓ to zero and considering

$$\tilde{A} = \sum_{k=1}^{\ell} \sigma_k u_k v_k^T.$$

Such a low-rank approximation may be found in *principal component analysis* (PCA) with ℓ first principal components considered. PCA applies SVD to the data matrix after certain preprocessing (centralization or standardization of data samples) is performed. We refer the reader to a linear algebra text [17] for more theoretical consideration of the SVD properties and algorithms.

One may relate biclustering to the SVD via consideration an idealized data matrix. If the data matrix has a block-diagonal structure (with all elements outside the blocks equal to zero), *it is natural to associate each block with a bicluster*. On the other hand, it is easy to see that each pair of singular vectors will designate one such bicluster by nonzero components in the vector. More precisely, if the data matrix is of the form

$$A = \begin{pmatrix} A_1 & 0 & \dots & 0 \\ 0 & A_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_r \end{pmatrix},$$

where $\{A_k\}$, $k = 1, \dots, r$ are arbitrary matrices, then for each A_k there will be a singular vector pair (u_k, v_k) such that nonzero components of u_k correspond to rows occupied by A_k and nonzero components of v_k correspond to columns

occupied by A_k . In a less idealized case, when the elements outside the diagonal blocks are not necessarily zeros but diagonal blocks still contain dominating values, the SVD is able to reveal the biclusters too as dominating components in the singular vector pairs.

Hence, the SVD represents a handy tool for biclustering algorithms. Below we show that many biclustering methods either use the SVD directly or have a certain association with the SVD concept. Furthermore, we believe that the introduction into SVD must precede any discussion on biclustering methods because its theoretical value for simultaneous analysis of samples and features of data cannot be overestimated. As of now, not many biclustering methods are theoretically justified in sense that it is not clear what mathematical properties of sample/feature vectors would tie them together in a common bicluster in one or another biclustering routine. Next, it is not obvious whether results of two arbitrarily chosen biclustering methods applied to the same data matrix are expected to contradict each other or not, and, if the contradiction occurs, whether it is natural (e.g., two methods analyze completely different properties of data) or tells us that one method is more precise than the other. We see SVD as the tool for resolution of such issues as we may hope to bring biclustering to the common ground, for example, by relating each biclustering method to SVD after a certain algebraic transformation applied to the data.

5. Methods

5.1. “Direct clustering”

Apparently the earliest biclustering algorithm that may be found in the literature is so-called direct clustering by Hartigan [18] also known as *block clustering*. This approach relies on statistical analysis of submatrices to form the biclusters. Namely, the quality of a bicluster $(\mathcal{S}_k, \mathcal{F}_k)$ is assessed by the variance

$$\text{VAR}(\mathcal{S}_k, \mathcal{F}_k) = \sum_{i \in \mathcal{F}_k} \sum_{j \in \mathcal{S}_k} (a_{ij} - \mu_k)^2,$$

where μ_k is the average value in the bicluster:

$$\mu_k = \frac{\sum_{i \in \mathcal{F}_k} \sum_{j \in \mathcal{S}_k} a_{ij}}{|\mathcal{F}_k| |\mathcal{S}_k|}.$$

A bicluster is considered perfect if it has zero variance, so biclusters with lower variance are considered to be better than biclusters with higher variance. This, however, leads to an undesirable effect: single-row, single-column submatrices become ideal biclusters as their variance is zero. The issue is resolved by fixing the number of biclusters and minimizing the objective

$$\text{VAR}(\mathcal{S}, \mathcal{F}) = \sum_{k=1}^r \sum_{i \in \mathcal{F}_k} \sum_{j \in \mathcal{S}_k} (a_{ij} - \mu_k)^2.$$

Hartigan mentioned that other objective functions may be used to find biclusters with other desirable properties, e.g., minimizing variance in rows, variance in columns, or biclusters following certain patterns.

5.2. Node-deletion algorithm

A more sophisticated criterion for constructing patterned biclusters was introduced by Cheng and Church [19]. It is based on minimization of so-called *mean squared residue*. To formulate it, let us introduce the following notation. Let

$$\mu_{ik}^{(r)} = \frac{1}{|\mathcal{S}_k|} \sum_{j \in \mathcal{S}_k} a_{ij} \quad (2)$$

be the mean of the i th row in the sample cluster \mathcal{S}_k ,

$$\mu_{jk}^{(c)} = \frac{1}{|\mathcal{F}_k|} \sum_{i \in \mathcal{F}_k} a_{ij} \quad (3)$$

be the mean of the j th column in the feature cluster \mathcal{F}_k , and

$$\mu_k = \frac{\sum_{i \in \mathcal{F}_k} \sum_{j \in \mathcal{S}_k} a_{ij}}{|\mathcal{F}_k| |\mathcal{S}_k|}$$

be the mean value in the bicluster $(\mathcal{S}_k, \mathcal{F}_k)$. The *residue* of element a_{ij} is defined as

$$r_{ij} = a_{ij} - \mu_{ik}^{(r)} - \mu_{jk}^{(c)} + \mu_k, \quad (4)$$

$i \in \mathcal{F}_k, j \in \mathcal{S}_k$. Finally, the *mean squared residue* score of the bicluster $(\mathcal{S}_k, \mathcal{F}_k)$ is defined as

$$H_k = \sum_{i \in \mathcal{F}_k} \sum_{j \in \mathcal{S}_k} (r_{ij})^2.$$

This value is equal to zero if all columns of the bicluster are equal to each other (that would imply that all rows are equal too). A bicluster $(\mathcal{S}_k, \mathcal{F}_k)$ is called a δ -bicluster if $H_k \leq \delta$. Cheng and Church proved that finding the largest square δ -bicluster is *NP-hard*. So, they used a greedy procedure starting from the entire data matrix and successively removing columns or rows contributing most to the mean squared residue score. The brute-force deletion algorithm testing the deletion of each row and column would be still quite expensive in the sense of time complexity as it would require $O((m+n)mn)$ operations. However, the authors employed a simplified search for columns and rows to delete choosing a column with maximal

$$d(j) = \frac{1}{|\mathcal{F}_k|} \sum_{i \in \mathcal{F}_k} r_{ij}^2,$$

a row with maximal

$$d(i) = \frac{1}{|\mathcal{S}_k|} \sum_{j \in \mathcal{S}_k} r_{ij}^2,$$

or subsets of columns or rows for which $d(j)$ or $d(i)$ exceeds a certain threshold above the current mean square residue of the bicluster. They have proved that any such deletion can only decrease the current mean square residue. These deletions are performed until a δ -bicluster is obtained. Then, as the constructed co-cluster can be not maximal (i.e., some of the previously removed columns or rows can be added without violating the δ -bicluster condition), the authors used a column and row addition algorithm. Namely, they proved that adding any column (row) with $d(j)$ ($d(i)$) below the current mean square residue does not increase it. Therefore, successive addition of such columns and rows leads to a maximal δ -bicluster.

Software implementation of the method as well as some test data sets are available at [20].

Bryan et al. improved the node-deletion algorithm of Cheng and Church applying a simulated annealing technique. They reported a better performance on a variety of data sets in [21].

5.3. FLOC algorithm

Yang et al. generalized the definition of residue used in the node-deletion algorithm to allow missing data entries (i.e., some a_{ij} may be unknown) [22,23]. For a bicluster $(\mathcal{S}_k, \mathcal{F}_k)$, they introduced the notion of *alpha-occupancy* meaning that for each sample $j \in \mathcal{S}_k$ the number of known data entries $a_{ij}, i \in \mathcal{F}_k$ is greater than $\alpha|\mathcal{F}_k|$ and for each feature $i \in \mathcal{F}_k$ the number of known data entries $a_{ij}, j \in \mathcal{S}_k$ is greater than $\alpha|\mathcal{S}_k|$. They also defined the *volume* of a bicluster as the number of known data entries in the bicluster, and the average values $\mu_{ik}^{(r)}, \mu_{jk}^{(c)}$ and μ_k are calculated with respect to the known data entries only. The authors developed a heuristic algorithm FLOC (flexible overlapped clustering) to find r biclusters with low average residue. First, the biclusters are generated randomly with a chosen probability ρ for each sample and feature to be included in a bicluster. Then, for each feature and sample, and for each bicluster it is calculated how much the addition of this feature/sample (if it is currently not in the bicluster) or its removal (if it is currently in the bicluster) reduces the residue of the bicluster. If at least one of such actions reduces the residue, the one achieving the largest reduction is performed.

When no further residue reduction is possible, the method stops. It is easy to show that the computational complexity of the method is $O((m + n)mnrp)$, where p is the number of iterations till termination. The authors claim that in the computational experiments they performed p is of the order of 10.

The FLOC algorithm is also able to take into account various additional constraints on biclusters by eliminating certain feature/sample additions/removals from consideration.

5.4. Biclustering via spectral bipartite graph partitioning

In [24] Dhillon proposed the following method of biclustering. Represent each sample and each feature of a data set as a vertex of a graph $G(V, E)$, $|V| = m + n$. Between the vertex corresponding to sample $j = 1, \dots, n$ and the vertex corresponding to feature $i = 1, \dots, m$ introduce an edge with weight a_{ij} . The graph has no edges between vertices representing samples, as well as between vertices representing features. Thus, the graph is bipartite with \mathcal{F} and \mathcal{S} representing its color classes. The graph G has the following weighted adjacency matrix

$$M = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}. \tag{5}$$

Now, a partition of the set of vertices into r parts V_1, V_2, \dots, V_r ,

$$V = V_1 \cup V_2 \cup \dots \cup V_r,$$

$$V_k \cap V_\ell = \emptyset, \quad k \neq \ell, \quad k, \ell = 1, \dots, r,$$

will provide a biclustering of the data set. Define the cost of the partition as the total weight of edges cut by it:

$$\text{cut}(V_1, \dots, V_r) = \sum_{k=1}^{r-1} \sum_{\ell=k+1}^r \sum_{i \in V_k} \sum_{j \in V_\ell} m_{ij}. \tag{6}$$

When we are looking for a biclustering maximizing in-class expression values (thus creating dominating submatrices of biclusters) it is natural to seek minimization of the defined cut value. Besides, we should be looking for rather balanced in size biclusters as otherwise the cut value is most probably minimized with all but one biclusters containing one sample-feature pair only. This problem can be tackled with an SVD-related algorithm. Let us introduce the following:

Definition 2. The Laplacian matrix L_G of $G(V, E)$ is a $|V| \times |V|$ symmetric matrix, with one row and one column for each vertex, such that

$$L_{ij} = \begin{cases} \sum_k m_{ik} & \text{if } i = j, \\ -m_{ij} & \text{if } i \neq j \text{ and } (i, j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

Let a partition $V = V_1 \cup V_2$ of the graph be defined via a ± 1 vector $p = (p_i)_{i=1, \dots, |V|}$ such that

$$p_i = \begin{cases} +1, & i \in V_1, \\ -1, & i \in V_2. \end{cases}$$

The Laplacian matrix is connected to the weight of a cut through the following:

Theorem 1. Given the Laplacian matrix L_G of G and a partition vector p , the Rayleigh Quotient

$$\frac{p^T L p}{p^T p} = \frac{4}{|V|} \text{cut}(V_1, V_2).$$

By this theorem, the cut is obviously minimized with the trivial solution, i.e., when all p_i are either -1 or 1 . So, to achieve a balanced partition we need to modify the objective function. Let us assign a positive weight w_i to each vertex

$i \in V$, and let $W = \text{diag}(w_1, w_2, \dots, w_{|V|})$ be the diagonal matrix of these weights. We denote

$$\text{weight}(V_\ell) = \sum_{i \in V_\ell} w_i.$$

Now, the following objective function allows us to achieve balanced clusters:

$$Q(V_1, V_2) = \frac{\text{cut}(V_1, V_2)}{\text{weight}(V_1)} + \frac{\text{cut}(V_1, V_2)}{\text{weight}(V_2)}.$$

Let us denote $v_\ell = \sum_{i \in V_\ell} w_i$ and introduce the generalized partition vector with elements

$$q_i = \begin{cases} +\sqrt{\frac{v_2}{v_1}}, & i \in V_1, \\ -\sqrt{\frac{v_1}{v_2}}, & i \in V_2. \end{cases}$$

The following theorem generalizes Theorem 1.

Theorem 2.

$$\frac{q^T L q}{q^T W q} = \frac{\text{cut}(V_1, V_2)}{\text{weight}(V_1)} + \frac{\text{cut}(V_1, V_2)}{\text{weight}(V_2)}. \tag{7}$$

Minimizing expression (7) is NP-hard. However, a relaxed version of this problem can be solved via a generalized eigendecomposition (notice that $q^T W e = 0$).

Theorem 3. *The problem*

$$\begin{aligned} \min_{x \neq 0} \quad & \frac{x^T L x}{x^T W x} \\ \text{s.t.} \quad & x^T W e = 0, \end{aligned} \tag{8}$$

is solved when q is the eigenvector corresponding to the second smallest eigenvalue λ_2 of the generalized eigenvalue problem

$$Lz = \lambda Wz. \tag{9}$$

We can solve this problem for the bipartite graph case via the SVD. Choosing the weight matrix W to be equal to the degree matrix, we have

$$L = \begin{pmatrix} D_1 & -A \\ -A^T & D_2 \end{pmatrix}$$

and

$$W = \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix},$$

where D_1 and D_2 are diagonal matrices such that $D_1(i, i) = \sum_j a_{ij}$ and $D_2(j, j) = \sum_i a_{ij}$. Then (9) becomes

$$\begin{pmatrix} D_1 & -A \\ -A^T & D_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix},$$

or denoting $u = D_1^{1/2}x$ and $v = D_2^{1/2}y$,

$$D_1^{-1/2}AD_2^{-1/2}v = (1 - \lambda)u,$$

$$D_2^{-1/2}A^T D_1^{-1/2}u = (1 - \lambda)v,$$

which precisely defines the SVD of the normalized matrix $\hat{A} = D_1^{-1/2}AD_2^{-1/2}$. So, the balanced cut minimization problem can be solved by finding the second largest singular value of this normalized matrix and the singular vector pair corresponding to it that can be used to obtain the biclustering to two classes. In case of multiclass partitioning, Dhillon used $\ell = \lceil \log_2 r \rceil$ singular vectors $u_2, u_3, \dots, u_{\ell+1}$ and $v_2, v_3, \dots, v_{\ell+1}$ to form the ℓ -dimensional data set

$$Z = \begin{pmatrix} D_1^{-1/2}U \\ D_2^{-1/2}V \end{pmatrix},$$

where $U = (u_2, \dots, u_{\ell+1})$ and $V = (v_2, \dots, v_{\ell+1})$. After such a significant dimensionality reduction is performed, the rows of the matrix Z (which represent both samples and features of the original data set) are clustered with a simple k -means algorithm [6].

Dhillon reports encouraging computational results for text mining problems. Very similar spectral biclustering routines for microarray data have been suggested by Kluger et al. [25]. In addition to working with the singular vectors of \hat{A} , they considered two other normalization methods that can be used before applying the SVD. The first one is *bistochastization*. It makes all row sums equal and all column sums equal too (generally, to a different constant). It is known from Sinkhorn's theorem that under quite general conditions on the matrix A there exist diagonal matrices D_1 and D_2 such that D_1AD_2 achieves bistochastization [26]. The other approach is applicable if sample/feature subvectors within a bicluster are expected to be shifted by a constant with respect to each other (i.e., vectors a and b are considered similar if $a \approx b + \alpha e$, where α is the constant and e is the all-one vector). When similar data are expected to be scaled by different constants (i.e., $a \approx \alpha b$), the desirable property can be achieved by applying a logarithm to all data entries. Then, defining

$$\bar{a}_{i.} = \frac{1}{n} \sum_{j=1}^n a_{ij},$$

$$\bar{a}_{.j} = \frac{1}{m} \sum_{i=1}^m a_{ij},$$

and

$$\bar{a}_{..} = \frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n a_{ij},$$

the normalized data are obtained as

$$b_{ij} = a_{ij} - \bar{a}_{i.} - \bar{a}_{.j} + \bar{a}_{..}$$

After computing the singular vectors, it is decided which of them contain the relevant information about the optimal data partition. To extract partitioning information from the system of singular vectors, each of them is examined by fitting to a piecewise constant vector. That is, the entries of an eigenvector is sorted and all possible thresholds between classes are considered. Such a procedure is equivalent to searching for good optima in one-dimensional k -means problem. Then few best singular vectors can be selected to run k -means on the data projected onto them.

5.5. Matrix iteration algorithms for minimizing sum-squared residue

Cho et al. proposed a co-clustering algorithm minimizing the sum-squared residue throughout all co-clusters [27]. Thus, this approach does not take into account any correspondence between clusters of samples and clusters of features,

but considers all the submatrices formed by them. The algorithm is based on algebraic properties of the matrix of residues.

For a given clustering of features $(\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_q)$, introduce a feature cluster indicator matrix $F = (f_{ik})_{m \times q}$ such that $f_{ik} = |\mathcal{F}_k|^{-1/2}$ if $i \in \mathcal{F}_k$ and $f_{ik} = 0$ otherwise. Also, for a given clustering of samples $(\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_r)$, introduce a sample cluster indicator matrix $S = (s_{jk})_{n \times r}$ such that $s_{jk} = |\mathcal{S}_k|^{-1/2}$ if $j \in \mathcal{S}_k$ and $s_{jk} = 0$ otherwise. Notice that these matrices are orthonormal, that is, all columns are orthogonal to each other and have unit length. Now, let $H = (h_{ij})_{m \times n}$ be the residue matrix. There are two choices for h_{ij} definition. It may be defined similar to (4):

$$h_{ij} = a_{ij} - \mu_{ik}^{(r)} - \mu_{j\ell}^{(c)} + \mu_{k\ell}, \quad (10)$$

where $i \in \mathcal{F}_\ell$, $j \in \mathcal{S}_k$, $\mu^{(r)}$ and $\mu^{(c)}$ are defined as in (2) and (3), and $\mu_{k\ell}$ is the average of the co-cluster $(\mathcal{S}_k, \mathcal{F}_\ell)$:

$$\mu_{k\ell} = \frac{\sum_{i \in \mathcal{F}_\ell} \sum_{j \in \mathcal{S}_k} a_{ij}}{|\mathcal{F}_\ell| |\mathcal{S}_k|}.$$

Alternatively, h_{ij} may be defined just as the difference between a_{ij} and the co-cluster average:

$$h_{ij} = a_{ij} - \mu_{k\ell}. \quad (11)$$

By direct algebraic manipulations it can be shown that

$$H = A - FF^T ASS^T \quad (12)$$

in case of (11) and

$$H = (I - FF^T)A(I - SS^T) \quad (13)$$

in case of (10).

The method tries to minimize $\|H\|^2$ using an iterative process such that on each iteration a current co-clustering is updated so that $\|H\|^2$, at least, does not increase. The authors point out that finding the global minimum for $\|H\|^2$ over all possible co-clusterings would lead to an NP-hard problem. There are two types of clustering updates used: batch (when all samples or features may be moved between clusters at one time) and incremental (one sample or one feature is moved at a time). In case of (11) the batch algorithm works as follows:

Algorithm 1 (Coclus_{H1}).

Input: data matrix A , number of sample clusters r , number of feature clusters q .

Output: clustering indicators S and F .

1. Initialize S and F .

2. $objval := \|A - FF^T ASS^T\|^2$.

3. $\Delta := 1$, $\tau := 10^{-2} \|A\|^2 \{Adjustable\}$.

4. While $\Delta > \tau$:

4.1. $A^S := FF^T AS$;

4.2. for $j := 1$ to n assign j th sample to cluster \mathcal{S}_k with smallest $\|A_{.j} - |\mathcal{S}_k|^{-1/2} A_{.k}^S\|^2$;

4.3. update S with respect to the new clustering;

4.4. $A^F := F^T ASS^T$;

4.5. for $i := 1$ to m assign i th feature to cluster \mathcal{F}_k with smallest $\|A_{i.} - |\mathcal{F}_k|^{-1/2} A_{i.}^F\|^2$;

4.6. update F with respect to the new clustering;

4.7. $oldobj := objval$, $objval := \|A - FF^T ASS^T\|^2$;

4.8. $\Delta := |oldobj - objval|$.

5. STOP.

In case of (10) the algorithm is similar but uses a bit different matrix manipulations:

Algorithm 2 (*Coclus_{H2}*).

Input: data matrix A , number of sample clusters r , number of feature clusters q .

Output: clustering indicators S and F .

1. Initialize S and F .

2. $objval := \|(I - FF^T)A(I - SS^T)\|^2$.

3. $\Delta := 1, \tau := 10^{-2}\|A\|^2\{\text{Adjustable}\}$.

4. While $\Delta > \tau$:

4.1. $A^S := (I - FF^T)AS, A^P := (I - FF^T)A$;

4.2. for $j := 1$ to n assign j th sample to cluster \mathcal{S}_k with smallest $\|A_{j.}^P - |\mathcal{S}_k|^{-1/2}A_{k.}^S\|^2$;

4.3. update S with respect to the new clustering;

4.4. $A^F := F^T A(I - SS^T), A^P := A(I - SS^T)$;

4.5. for $i := 1$ to m assign i th feature to cluster \mathcal{F}_k with smallest $\|A_{i.}^P - |\mathcal{F}_k|^{-1/2}A_{k.}^F\|^2$;

4.6. update F with respect to the new clustering;

4.7. $oldobj := objval, objval := \|(I - FF^T)A(I - SS^T)\|^2$;

4.8. $\Delta := |oldobj - objval|$.

5. STOP.

To describe the incremental algorithm, we first note that in case of (10) H is defined as in (13), and minimization of $\|H\|^2$ is equivalent to maximization of $\|F^T AS\|^2$. So, suppose we would like to improve the objective function by moving a sample from cluster \mathcal{S}_k to cluster $\mathcal{S}_{k'}$. Denote $F^T A$ by \bar{A} and the new sample clustering indicator matrix by \tilde{S} . As S and \tilde{S} differ only in columns k and k' , the objective can be rewritten as

$$\|\bar{A}\tilde{S}_{.k'}\|^2 - \|\bar{A}S_{.k'}\|^2 + \|\bar{A}\tilde{S}_{.k}\|^2 - \|\bar{A}S_{.k}\|^2. \quad (14)$$

So, the inner loop of the incremental algorithm looks through all possible one sample moves and chooses the one increasing (14) most. A similar expression can be derived for features. Next, it can be shown that in case (11) when H is defined as in (12), the objective can be reduced to

$$\|A\tilde{S}_{.k'}\|^2 - \|AS_{.k'}\|^2 + \|A\tilde{S}_{.k}\|^2 - \|AS_{.k}\|^2 - \|\bar{A}\tilde{S}_{.k'}\|^2 + \|\bar{A}S_{.k'}\|^2 - \|\bar{A}\tilde{S}_{.k}\|^2 + \|\bar{A}S_{.k}\|^2, \quad (15)$$

so the incremental algorithm just uses (15) instead of (14).

Notice the direct relation of the method to the SVD. Maximization of $\|F^T AS\|^2$ if F and S were just constrained to be orthonormal matrices would be solved by $F = U$ and $S = V$, where U and V are as in (1). F and S have the additional constraint on the structure (being a clustering indicator). However, the SVD helps to initialize the clustering indicator matrices and provides a lower bound on the objective (as the sum of squares of the singular values).

Software with the implementation of both cases of this method is available at [28].

5.6. Double conjugated clustering

Double conjugated clustering (DCC) is a node-driven biclustering technique that can be considered a further development of such clustering methods as k -means [6] and Self-Organizing Maps (SOM) [7]. The method was developed by Busygin et al. [29]. It operates in two spaces—space of samples and space of features—applying in each of them either k -means or SOM training iterations. Meanwhile, after each one-space iteration its result updates the other map of clusters by means of a matrix projection. The method works as follows.

Introduce a matrix $C = (c_{ik})_{m \times r}$ which will be referred to as *samples nodes* or *samples map* and a matrix $D = (d_{jk})_{n \times r}$ which will be referred to as *features nodes* or *features map*. This designates r nodes for samples and r nodes for features that will be used for one-space clustering iterations such as k -means or SOM (in the latter case, the nodes are to be arranged with respect to a certain topology that will determine node neighborhoods). We start from the samples map, initialize it with random numbers and perform a one-space clustering iteration (for instance, in case of k -means we

assign each sample to closest node and then update each node storing in it the centroid of the assigned samples). Now the content of C is projected to form D with a matrix transformation:

$$D := \mathcal{B}(A^T C),$$

where $\mathcal{B}(M)$ is the operator normalizing each column of matrix M to the unit length. The matrix multiplication that transforms nodes of one space to the other can be justified with the following argument. The value c_{ik} is the weight of i th feature in the k th node. So, the k th node of the features map is constructed as a linear combination of the features such that c_{ik} is the coefficient of the i th feature in it. The unit normalization keeps the magnitude of node vectors constrained. Next, after the projection, the features map is updated with the similar one-space clustering iteration, and then the backwards projection is applied:

$$C := \mathcal{B}(AD),$$

which is justified in the similar manner using the fact that d_{jk} is the weight of the j th sample in the k th node. This cycle is repeated until no samples and features are moved anymore, or stops after a predefined number of iterations.

To be consistent with unit normalization of the projected nodes, the authors have chosen to use cosine metrics for one-space iterations, which is not affected by differences in magnitudes of the clustered vectors. This also prevents undesirable clustering of all low-magnitude elements into a single cluster that often happens when a node-driven clustering is performed using the Euclidean metric.

The DCC method has a close connection to the SVD that can be observed in its computational routine. Notice that if one “forgets” to perform the one-space clustering iterations, then DCC executes nothing else but the power method for the SVD [17]. In such case all samples nodes would converge to the dominating left singular vector and all features nodes would converge to the dominating right singular vector of the data matrix. However, the one-space iterations prevent this from happening moving the nodes towards centroids of different sample/feature clusters. This acts similarly to re-orthogonalization in the power method when not only the dominating but also a bunch of next singular vector pairs are sought. This way DCC can be seen as an alteration of the power method for SVD relaxing the orthogonality requirement for the iterated vectors but making them more appealing to groups of similar samples/features of the data.

5.7. Consistent biclustering via fractional 0–1 programming

Let each sample be already assigned somehow to one of the classes $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_r$. Introduce a 0–1 matrix $S = (s_{jk})_{n \times r}$ such that $s_{jk} = 1$ if $j \in \mathcal{S}_k$, and $s_{jk} = 0$ otherwise. The sample class centroids can be computed as the matrix $C = (c_{ik})_{m \times r}$:

$$C = AS(S^T S)^{-1}, \tag{16}$$

whose k th column represents the centroid of the class \mathcal{S}_k .

Consider a row i of the matrix C . Each value in it gives us the average expression of the i th feature in one of the sample classes. As we want to identify the checkerboard pattern in the data, we have to assign the feature to the class where it is most expressed. So, let us classify the i th feature to the class \hat{k} with the maximal value $c_{i\hat{k}}$:

$$i \in \mathcal{F}_{\hat{k}} \Rightarrow \forall k = 1, \dots, r, \quad k \neq \hat{k} : c_{i\hat{k}} > c_{ik}. \tag{17}$$

Now, provided the classification of all features into classes $\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_r$, let us construct a classification of samples using the same principle of maximal average expression and see whether we will arrive at the same classification as the initially given one. To do this, construct a 0–1 matrix $F = (f_{ik})_{m \times r}$ such that $f_{ik} = 1$ if $i \in \mathcal{F}_k$ and $f_{ik} = 0$ otherwise. Then, the feature class centroids can be computed in form of matrix $D = (d_{jk})_{n \times r}$:

$$D = A^T F(F^T F)^{-1}, \tag{18}$$

whose k th column represents the centroid of the class \mathcal{F}_k . The condition on sample classification we need to verify is

$$j \in \mathcal{S}_{\hat{k}} \Rightarrow \forall k = 1, \dots, r, \quad k \neq \hat{k} : d_{j\hat{k}} > d_{jk}. \tag{19}$$

The framework for feature selection and supervised biclustering proposed in [30,76] is based on the following definition:

Definition 3. A biclustering \mathcal{B} will be called *consistent* if both relations (17) and (19) hold, where the matrices C and D are defined as in (16) and (18).

In contrast to other biclustering schemes, this definition of consistent biclustering is justified by the fact that *consistent biclustering implies separability of the classes by convex cones* [30]:

Theorem 4. Let \mathcal{B} be a consistent biclustering. Then there exist convex cones $\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_r \subseteq \mathbb{R}^m$ such that all samples from \mathcal{S}_k belong to the cone \mathcal{P}_k and no other sample belongs to it, $k = 1, \dots, r$.

Similarly, there exist convex cones $\mathcal{Q}_1, \mathcal{Q}_2, \dots, \mathcal{Q}_r \subseteq \mathbb{R}^n$ such that all features from \mathcal{F}_k belong to the cone \mathcal{Q}_k and no other feature belongs to it, $k = 1, \dots, r$.

It also follows from the conic separability that convex hulls of classes are separated, i.e., they do not intersect.

We also say that a data set is *biclustering-admitting* if some consistent biclustering for it exists. Furthermore, the data set will be called *conditionally biclustering-admitting* with respect to a given (partial) classification of some samples and/or features if there exists a consistent biclustering preserving the given (partial) classification.

Assuming that we are given the training set $A = (a_{ij})_{m \times n}$ with the classification of samples into classes $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_r$, we are able to construct the corresponding classification of features according to (17). Next, if the obtained biclustering is not consistent, our goal is to exclude some features from the data set so that the biclustering with respect to the residual feature set is consistent.

Formally, let us introduce a vector of 0–1 variables $x = (x_i)_{i=1, \dots, m}$ and consider the i th feature selected if $x_i = 1$. The condition of biclustering consistency (19), when only the selected features are used, becomes

$$\frac{\sum_{i=1}^m a_{ij} f_{i\hat{k}} x_i}{\sum_{i=1}^m f_{i\hat{k}} x_i} > \frac{\sum_{i=1}^m a_{ij} f_{ik} x_i}{\sum_{i=1}^m f_{ik} x_i} \quad \forall j \in \mathcal{S}_{\hat{k}}, \hat{k}, k = 1, \dots, r, \hat{k} \neq k. \tag{20}$$

We will use the fractional relations (20) as constraints of an optimization problem selecting the feature set. It may incorporate various objective functions over x , depending on the desirable properties of the selected features, but one general choice is to select the maximal possible number of features in order to lose minimal amount of information provided by the training set. In this case, the objective function is

$$\max \sum_{i=1}^m x_i. \tag{21}$$

The optimization problem (21), (20) is a specific type of *fractional 0–1 programming problem*, which can be solved using the approach described in [30].

Moreover, we can strengthen the class separation by introduction of a coefficient greater than 1 for the right-hand side of inequality (20). In this case, we improve the quality of the solution modifying (20) as

$$\frac{\sum_{i=1}^m a_{ij} f_{i\hat{k}} x_i}{\sum_{i=1}^m f_{i\hat{k}} x_i} \geq (1 + t) \frac{\sum_{i=1}^m a_{ij} f_{ik} x_i}{\sum_{i=1}^m f_{ik} x_i} \tag{22}$$

for all $j \in \mathcal{S}_{\hat{k}}, \hat{k}, k = 1, \dots, r, \hat{k} \neq k$, and $t > 0$ is a constant that becomes a parameter of the method, which we call *the parameter of separation*.

After the feature selection is done, we perform classification of test samples according to (19). That is, if $b = (b_i)_{i=1, \dots, m}$ is a test sample, we assign it to the class $\mathcal{S}_{\hat{k}}$ satisfying

$$\frac{\sum_{i=1}^m b_i f_{i\hat{k}} x_i}{\sum_{i=1}^m f_{i\hat{k}} x_i} \geq \frac{\sum_{i=1}^m b_i f_{ik} x_i}{\sum_{i=1}^m f_{ik} x_i} \tag{23}$$

for all $k = 1, \dots, r, \hat{k} \neq k$.

5.8. Information-theoretic based co-clustering

In this method, developed by Dhillon et al. in [31], we treat the input data set $(a_{ij})_{m \times n}$ as a joint probability distribution $p(X, Y)$ between two discrete random variables X and Y , which can take values in the sets $\{x_1, x_2, \dots, x_m\}$ and $\{y_1, y_2, \dots, y_n\}$, respectively.

Formally speaking, the goal of the proposed procedure is to cluster X into at most k disjoint clusters $\hat{X} = \{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_k\}$ and Y into at most l disjoint clusters $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_l\}$. Put differently, we are looking for mappings C_X and C_Y such that

$$C_X : \{x_1, x_2, \dots, x_m\} \rightarrow \{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_k\},$$

$$C_Y : \{y_1, y_2, \dots, y_n\} \rightarrow \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_l\},$$

i.e., $\hat{X} = C_X(X)$ and $\hat{Y} = C_Y(Y)$, and a tuple (C_X, C_Y) is referred to as co-clustering.

Before we proceed with a description of the technique let us recall some definitions from probability and information theory.

The *relative entropy*, or the *Kullback–Leibler (KL) divergence* between two probability distributions $p_1(x)$ and $p_2(x)$ is defined as

$$D(p_1 \| p_2) = \sum_x p_1(x) \log \frac{p_1(x)}{p_2(x)}.$$

Kullback–Leibler divergence can be considered as a “distance” of a “true” distribution p_1 to an approximation p_2 .

The *mutual information* $I(X; Y)$ of two random variables X and Y is the amount of information shared between these two variables. In other words, $I(X; Y) = I(Y; X)$ measures how much X tells about Y and, vice versa, how much Y tells about X . It is defined as

$$I(X; Y) = \sum_y \sum_x p(x, y) \log \frac{p(x, y)}{p(x)p(y)} = D(p(x, y) \| p(x)p(y)).$$

Now, we are looking for an optimal co-clustering, which minimizes *the loss in mutual information*

$$\min_{\hat{X}, \hat{Y}} I(X; Y) - I(\hat{X}, \hat{Y}). \tag{24}$$

Define $q(X, Y)$ to be the following distribution

$$q(x, y) = p(\hat{x}, \hat{y})p(x|\hat{x})p(y|\hat{y}), \tag{25}$$

where $x \in \hat{x}$ and $y \in \hat{y}$. Obviously, $p(x|\hat{x}) = p(x)/p(\hat{x})$ if $\hat{x} = C_X(x)$ and 0, otherwise.

The following result states an important relation between the loss of information and distribution $q(X, Y)$ [32]:

Lemma 1. For a fixed co-clustering (C_X, C_Y) , we can write the loss in mutual information as

$$I(X; Y) - I(\hat{X}; \hat{Y}) = D(p(X, Y) \| q(X, Y)). \tag{26}$$

In other words, finding an optimal co-clustering is equivalent to finding a distribution q defined by (25), which is close to p in KL divergence.

Consider the joint distribution of X, Y, \hat{X} and \hat{Y} denoted by $p(X, Y, \hat{X}, \hat{Y})$. Following the above lemma and (25) we are looking for a distribution $q(X, Y, \hat{X}, \hat{Y})$, an approximation of $p(X, Y, \hat{X}, \hat{Y})$, such that:

$$q(x, y, \hat{x}, \hat{y}) = p(\hat{x}, \hat{y})p(x|\hat{x})p(y|\hat{y}),$$

and $p(X, Y)$ and $q(X, Y)$ are considered as two-dimensional marginals of $p(X, Y, \hat{X}, \hat{Y})$ and $q(X, Y, \hat{X}, \hat{Y})$, respectively. The next lemma lies in the core of the proposed algorithm from [31].

Lemma 2. *The loss in mutual information can be expressed as*

(i) *a weighted sum of the relative entropies between row distributions $p(Y|x)$ and “row-lumped” distributions $q(Y|\hat{x})$,*

$$D(p(X, Y, \hat{X}, \hat{Y})\|q(X, Y, \hat{X}, \hat{Y})) = \sum_{\hat{x}} \sum_{x:C_X(x)=\hat{x}} p(x)D(p(Y|x)\|q(Y|\hat{x})),$$

(ii) *a weighted sum of the relative entropies between column distributions $p(X|y)$ and “column-lumped” distributions $q(X|\hat{y})$, that is,*

$$D(p(X, Y, \hat{X}, \hat{Y})\|q(X, Y, \hat{X}, \hat{Y})) = \sum_{\hat{y}} \sum_{y:C_Y(y)=\hat{y}} p(y)D(p(X|y)\|q(X|\hat{y})).$$

Due to Lemma 2 the objective function can be expressed only in terms of the row-clustering, or column-clustering. Starting with some initial co-clustering (C_X^0, C_Y^0) (and distribution q^0) we iteratively obtain new co-clusterings $(C_X^1, C_Y^1), (C_X^2, C_Y^2), \dots$, using column-clustering in order to improve row-clustering as

$$C_X^{t+1}(x) = \arg \min_{\hat{x}} D(p(Y|x)\|q^t(Y|\hat{x})), \tag{27}$$

and vice versa, using row-clustering to improve column-clustering as

$$C_Y^{t+2}(y) = \arg \min_{\hat{y}} D(p(X|y)\|q^t(X|\hat{y})). \tag{28}$$

Obviously, after each step (27), or (28) we need to recalculate the necessary distributions q^{t+1} and q^{t+2} . It can be proved that the described algorithm monotonically decreases the objective function (24), though it may converge only to a local minimum [31].

Software with the implementation of this method is available at [28].

In [32] the described alternating minimization scheme was generalized for Bregman divergences, which includes KL-divergence and Euclidean distance as special cases.

5.9. Biclustering via Gibbs sampling

The Bayesian framework can be a powerful tool to tackle problems involving uncertainty and noisy patterns. Thus it comes as a natural choice to apply it to data mining problems such as biclustering. Sheng et al. proposed a Bayesian technique for biclustering based on a simple frequency model for the expression pattern of a bicluster and on Gibbs sampling for parameter estimation [33]. This approach not only finds samples and features of a bicluster but also represents the pattern of a bicluster as a probabilistic model defined by the posterior distribution for the data values within the bicluster. The choice of Gibbs sampling also helps to avoid local minima in the expectation–maximization procedure that is used to obtain and adjust the probabilistic model.

Gibbs sampling is a well-known Markov chain Monte Carlo method [34]. It is used to sample random variables (x_1, x_2, \dots, x_k) when their marginal distribution of the joint distribution are too complex to sample directly from, but the conditional distributions can be easily sampled. Starting from initial values $(x_1^{(0)}, x_2^{(0)}, \dots, x_k^{(0)})$, the Gibbs samples draws values of the variables from the conditional distributions:

$$x_i^{(t+1)} \sim p(x_i|x_1^{(t+1)}, \dots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \dots, x_k^{(t)}),$$

$i = 1, \dots, k, t = 0, 1, 2, \dots$. It can be shown that the distribution of $(x_1^{(t)}, x_2^{(t)}, \dots, x_k^{(t)})$ converges to the true joint distribution $p(x_1, x_2, \dots, x_k)$ and the distributions of sequences $\{x_1^{(t)}\}, \{x_2^{(t)}\}, \dots, \{x_k^{(t)}\}$ converge to true marginal distribution of the corresponding variables.

The biclustering method works with $m + n$ 0–1 values $f = (f_i)_{i=1, \dots, m}$ (for features) and $s = (s_j)_{j=1, \dots, n}$ (for samples) indicating which features and samples are selected to the bicluster. These indicators are considered Bernoulli random variables with parameters λ_f and λ_s , respectively. The data are discretized and modeled with

multinomial distributions. The *background data* (i.e., all the data that do not belong to the bicluster) are considered to follow one single distribution $\phi = (\phi_1, \phi_2, \dots, \phi_\ell)$, $0 \leq \phi_k \leq 1$, $\sum_k \phi_k = 1$, $k = 1, \dots, \ell$, where ℓ is the total number of bins used for discretization. It is assumed that within the bicluster all features should behave similarly, but the samples are allowed to have different expression levels. That is, for data values of each sample j within the bicluster we assume a different distribution $(\theta_{1j}, \theta_{2j}, \dots, \theta_{\ell j})$, $0 \leq \theta_{kj} \leq 1$, $\sum_k \theta_{kj} = 1$, $k = 1, \dots, \ell$, and it is independent from the other samples. The probabilities λ_f , λ_s , $\{\phi_k\}$ and $\{\theta_{kj}\}$ are parameters of this Bayesian model, and therefore we need to include in the model their conjugate priors. Typically for Bayesian models, one chooses Beta distribution for the conjugate priors of Bernoulli random variables and Dirichlet distribution for the conjugate priors of multinomial random variables:

$$\phi \sim \text{Dirichlet}(\alpha),$$

$$\theta_{.j} \sim \text{Dirichlet}(\beta_j),$$

$$\lambda_f = \text{Beta}(\xi_f), \quad \lambda_s = \text{Beta}(\xi_s),$$

where α and β_j are parameter vectors of the Dirichlet distributions, and ξ_f and ξ_s are parameter vectors of the Beta distributions.

Denote the subvector of s with j th component removed by $s_{\bar{j}}$ and the subvector of f with i th component removed by $f_{\bar{i}}$. To derive the full conditional distributions, one can use the relations between distributions

$$p(f_i | f_{\bar{i}}, s, D) \propto p(f_i, f_{\bar{i}}, s, D) = p(f, s, D),$$

and

$$p(s_j | f, s_{\bar{j}}, D) \propto p(f, s_j, s_{\bar{j}}, D) = p(f, s, D),$$

where D is the observed discretized data. The distribution $p(f, s, D)$ can be obtained by integrating θ , ϕ , λ_f and λ_s out of the likelihood function $L(\theta, \phi, \lambda_f, \lambda_s | f, s, D)$:

$$L(\theta, \phi, \lambda_f, \lambda_s | f, s, D) = p(f, s, D | \theta, \phi, \lambda_f, \lambda_s) = p(D | f, s, \theta, \phi) p(f | \lambda_f) p(s | \lambda_s).$$

Using these conditional probabilities, we can perform the biclustering with the following algorithm:

Algorithm 3 (*Gibbs biclustering*).

1. Initialize randomly vectors f and s .
2. For each feature $i = 1, \dots, m$:
 - 2.1. Calculate $p_i = p(f_i = 1 | f_{\bar{i}}, s, D)$;
 - 2.2. Assign $f_i := 1$ with probability p_i or $f_i := 0$ otherwise.
3. For each sample $j = 1, \dots, n$:
 - 3.1. Calculate $p_j = p(s_j = 1 | f, s_{\bar{j}}, D)$;
 - 3.2. Assign $s_j := 1$ with probability p_j or $s_j := 0$ otherwise.
4. Repeat Steps 2–4 a predetermined number of iterations.

To obtain the biclustering, the probabilities p_i 's and p_j 's are averaged over all iterations and a feature/sample is selected in the bicluster if the average probability corresponding to it is above a certain threshold. More than one bicluster can be constructed by repeating the procedure while the probabilities corresponding to previously selected samples and features are permanently assigned to zero.

5.10. Statistical-algorithmic method for bicluster analysis (SAMBA)

Consider a bipartite graph $G(\mathcal{F}, \mathcal{S}, E)$, where the set of data features \mathcal{F} and the set of data samples \mathcal{S} form two independent sets, and there is an edge $(i, j) \in E$ between each feature i and each sample j iff the expression level of feature i changes significantly in sample j . Obviously, a bicluster $\mathcal{B}_0 = (\mathcal{S}_0, \mathcal{F}_0)$ should correspond to a subgraph

$H(\mathcal{F}_0, \mathcal{S}_0, E_0)$ of G . Next assign some weights to the edges and nonedges of G in such a way that the statistical significance of a bicluster matches the weight of the respective subgraph. Hence, in this setup biclustering is reduced to a search for *heavy* subgraphs in G . This idea is a cornerstone of the statistical-algorithmic method for bicluster analysis (SAMBA) developed by Tanay et al. [35,36]. Some additional details on construction of a bipartite graph $G(\mathcal{F}, \mathcal{S}, E)$ corresponding to features and samples can be found in the supporting information of [37].

The idea behind one of the possible schemes for edges' weight assignment from [36] works as follows. Let $p_{f,s}$ be the fraction of bipartite graphs with the degree sequence same as in G such that the edge $(f, s) \in E$. Suppose that the occurrence of an edge (f, s) is an independent Bernoulli random variable with parameter $p_{f,s}$. In this case, the probability of observing subgraph H is given by

$$p(H) = \left(\prod_{(f,s) \in E_0} p_{f,s} \right) \cdot \left(\prod_{(f,s) \notin E_0} (1 - p_{f,s}) \right). \tag{29}$$

Next consider another model, where edges between vertices from different partitions of a bipartite graph G occur independently with constant probability $p_c > \max_{(f,s) \in (\mathcal{F}, \mathcal{S})} p_{(f,s)}$. Assigning weights $\log(p_c/p_{(f,s)})$, to edges $(f, s) \in E_0$ and $\log(1 - p_c)/(1 - p_{(f,s)})$ to $(f, s) \notin E_0$ we can observe that the log-likelihood ratio for a subgraph H

$$\log L(H) = \sum_{(f,s) \in E_0} \frac{p_c}{p_{f,s}} + \sum_{(f,s) \notin E_0} \frac{1 - p_c}{1 - p_{f,s}} \tag{30}$$

is equal to the weight of the subgraph H . If we assume that we are looking for biclusters with the features behaving similarly within the set of samples of the respective bicluster then heavy subgraphs should correspond to “good” biclusters.

In [36] the algorithm for finding heavy subgraphs (biclusters) is based on the procedure for solving *the maximum bounded biclique problem*. In this problem we are looking for a maximum weight biclique in a bipartite graph $G(\mathcal{F}, \mathcal{S}, E)$ such that the degree of every feature vertex $f \in \mathcal{F}$ is at most d . It can be shown that maximum bounded biclique can be solved in $O(n2^d)$ time. At the first step of SAMBA for each vertex $f \in \mathcal{F}$ we find k heaviest bicliques containing f . During the next phase of the algorithm we try to improve the weight of the obtained subgraphs (biclusters) using a simple local search procedure. Finally, we greedily filter out biclusters with more than $L\%$ overlap.

SAMBA implementation is available as a part of EXPANDER, gene expression analysis and visualization tool, at [38].

5.11. Coupled two-way clustering

Coupled two-way clustering (CTWC) is a framework that can be used to build a biclustering on the basis of any one-way clustering algorithm. It was introduced by Getz et al. [39]. The idea behind the method is to find stable clusters of samples and features such that using one of the feature clusters results in stable clustering for samples and vice versa. The iterative procedure runs as follows. Initially, the entire set of samples \mathcal{S}_0^0 and the entire set of features \mathcal{F}_0^0 are considered stable clusters. \mathcal{F}_0^0 is used to cluster samples and \mathcal{S}_0^0 is used to cluster features. Denote by $\{\mathcal{F}_i^1\}$ and $\{\mathcal{S}_j^1\}$ the obtained clusters (which are considered stable with respect to \mathcal{F}_0^0 and \mathcal{S}_0^0). Now every pair $(\mathcal{F}_i^s, \mathcal{S}_j^t)$, $t, s = \{0, 1\}$ corresponds to a data submatrix, which can be clustered in the similar two-way manner to obtain clusters of the second order $\{\mathcal{F}_i^2\}$ and $\{\mathcal{S}_j^2\}$. Then again the process is repeated with each pair $(\mathcal{F}_i^s, \mathcal{S}_j^t)$ not used earlier to obtain the clusters on the next order, and so on until no new cluster satisfying certain criteria is obtained. The used criteria can impose constraints on cluster size, some statistical characteristics, etc.

Though any one-way clustering algorithm can be used within the described iterative two-way clustering procedure, the authors chose a hierarchical clustering method SPC [40,41]. The justification of this choice comes from the natural measure of relative cluster stability delivered by SPC. The SPC method originates from a physical model associating a break up of a cluster with a certain temperature at which this cluster loses stability. Therefore, it is easy to designate more stable clusters as those requiring higher temperature for further partitioning.

Online implementation of CTWC is available at [42].

5.12. Plaid models

Consider the perfect idealized biclustering situation. We have K biclusters along the main diagonal of the data matrix $A = (a_{ij})_{m \times n}$ with the same values of a_{ij} in each bicluster $k, k = 1, \dots, K$:

$$a_{ij} = \mu_0 + \sum_{k=1}^K \mu_k \rho_{ik} \kappa_{jk}, \tag{31}$$

where μ_0 is some constant value (“background color”), $\rho_{ik} = 1$ if feature i belongs to bicluster k ($\rho_{ik} = 0$, otherwise), $\kappa_{jk} = 1$ if sample j belongs to bicluster k ($\kappa_{jk} = 0$, otherwise) and μ_k is the value, which corresponds to bicluster k (“color” of bicluster k), i.e., $a_{ij} = \mu_0 + \mu_k$ if feature i and sample j belongs to the same bicluster k . We also require that each feature and sample must belong to exactly one bicluster, that is,

$$\forall i \sum_{k=1}^K \rho_{ik} = 1 \quad \text{and} \quad \forall j \sum_{k=1}^K \kappa_{jk} = 1, \tag{32}$$

respectively.

In [43] Lazeroni and Owen introduced a more complicated *plaid model* as a natural generalization of idealization (31)–(32). In this model, biclusters are allowed to overlap, and are referred to as *layers*. The values of a_{ij} in each layer are represented as

$$a_{ij} = \theta_{ij0} + \sum_{k=1}^K \theta_{ijk} \rho_{ik} \kappa_{jk}, \tag{33}$$

where the value of θ_{ij0} corresponds to a *background layer* and θ_{ijk} can be expressed as $\mu_k, \mu_k + \alpha_{ik}, \mu_k + \beta_{jk}$, or $\mu_k + \alpha_{ik} + \beta_{jk}$ depending on a particular situation.

We are looking for a plaid model such that the following objective function is minimized:

$$\min \sum_{i=1}^m \sum_{j=1}^n \left(a_{ij} - \theta_{ij0} - \sum_{k=1}^K \theta_{ijk} \rho_{ik} \kappa_{jk} \right)^2. \tag{34}$$

In [43] the authors developed a heuristic iterative-based algorithm for solving (34). Next we briefly describe the main idea of the approach. Suppose we have $K - 1$ layers and we are looking for the K th layer such that the objective function in (34) is minimized. Let

$$Z_{ij} = Z_{ij}^{K-1} = a_{ij} - \theta_{ij0} - \sum_{k=1}^{K-1} \theta_{ijk} \rho_{ik} \kappa_{jk}. \tag{35}$$

Substituting θ_{ijK} by $\mu_K + \alpha_{iK} + \beta_{jK}$, the objective function from (34) can be rewritten in terms of Z_{ij} as

$$\sum_{i=1}^m \sum_{j=1}^n (Z_{ij} - (\mu_K + \alpha_{iK} + \beta_{jK}) \rho_{ik} \kappa_{jk})^2. \tag{36}$$

Let $\rho_{iK}^{(0)}$ and $\kappa_{jK}^{(0)}$ be some starting values of our iteration algorithm. At each iteration step $s = 1, 2, \dots, S$ we update the values of $\rho_{iK}^{(s)}, \kappa_{jK}^{(s)}$ and $\theta_{ijK}^{(s)}$ applying the following simple procedure. The value of $\theta_{ijK}^{(s)}$ is obtained from $\rho_{iK}^{(s-1)}$ and $\kappa_{jK}^{(s-1)}$, then the values of $\rho_{iK}^{(s-1)}$ and $\kappa_{jK}^{(s-1)}$ are updated using $\theta_{ijK}^{(s)}$ and $\kappa_{jK}^{(s-1)}$, or $\theta_{ijK}^{(s)}$ and $\rho_{iK}^{(s-1)}$, respectively. Variables $\rho_{iK}^{(s)}$ and $\kappa_{jK}^{(s)}$ are relaxed, i.e., they can take values between 0 and 1. We fix them to be $\{0, 1\}$ during one of the last iterations of the algorithm.

More specifically, given ρ_{iK} and κ_{jK} , the value of $\theta_{ijK} = \mu_K + \alpha_{iK} + \beta_{jK}$ is updated as follows:

$$\mu_K = \frac{\sum_i \sum_j \rho_{iK} \kappa_{jK} Z_{ij}}{(\sum_i \rho_{iK}^2)(\sum_j \kappa_{jK}^2)},$$

$$\alpha_{iK} = \frac{\sum_j (Z_{ij} - \mu_K \rho_{iK} \kappa_{jK}) \kappa_{jK}}{\rho_{iK} \sum_j \kappa_{jK}^2}, \quad \beta_{jK} = \frac{\sum_i (Z_{ij} - \mu_K \rho_{iK} \kappa_{jK}) \rho_{iK}}{\kappa_{jK} \sum_i \rho_{iK}^2}.$$

Given θ_{ijK} and κ_{jK} , or θ_{ijK} and ρ_{iK} , we update ρ_{iK} , or κ_{jK} as

$$\rho_{iK} = \frac{\sum_j \theta_{ijK} \kappa_{jK} Z_{ij}}{\sum_j \theta_{ijK}^2 \kappa_{jK}^2} \quad \text{or} \quad \kappa_{jK} = \frac{\sum_i \theta_{ijK} \rho_{iK} Z_{ij}}{\sum_i \theta_{ijK}^2 \rho_{iK}^2},$$

respectively.

For more details of this technique including the selection of starting values $\rho_{iK}^{(0)}$ and $\kappa_{jK}^{(0)}$, stopping rules and other important issues we refer the reader to [43].

Software with the implementation of the discussed method is available at [44].

5.13. Order-preserving submatrix (OPSM) problem

In this model introduced by Ben-Dor et al. [45,46], given the data set $A = (a_{ij})_{m \times n}$, the problem is to identify a $k \times \ell$ submatrix (bicluster) $(\mathcal{F}_0, \mathcal{S}_0)$ such that the expression values of all features in \mathcal{F}_0 increase or decrease simultaneously within the set of samples \mathcal{S}_0 . In other words, in this submatrix we can find a permutation of columns such that in every row the values corresponding to selected columns are increasing. More formally, let \mathcal{F}_0 be a set of row indices $\{f_1, f_2, \dots, f_k\}$. Then there exists a permutation of \mathcal{S}_0 , which consists of column indices $\{s_1, s_2, \dots, s_\ell\}$, such that for all $i = 1, \dots, k$ and $j = 1, \dots, \ell - 1$ we have that

$$a_{f_i, s_j} < a_{f_i, s_{j+1}}.$$

In [45,46] it is proved that the **OPSM** problem is *NP*-hard. So, the authors designed a greedy heuristic algorithm for finding large order-preserving submatrices, which we briefly outline next.

Let $\mathcal{S}_0 \subset \{1, \dots, n\}$ be a set of column indices of size ℓ and $\pi = (s_1, s_2, \dots, s_\ell)$ be a linear ordering of \mathcal{S}_0 . The pair (\mathcal{S}_0, π) is called a *complete OPSM model*. A row $i \in \{1, \dots, m\}$ supports a complete model (\mathcal{S}_0, π) if

$$a_{i, s_1} < a_{i, s_2} < \dots < a_{i, s_\ell}.$$

For a complete model (\mathcal{S}_0, π) all supporting rows can be found in $O(nm)$ time.

A *partial model* $\theta = \{\langle s_1, \dots, s_c \rangle, \langle s_{\ell-d+1}, \dots, s_\ell \rangle, \ell\}$ of a complete model (\mathcal{S}_0, π) is given by the column indices of the c smallest elements $\langle s_1, \dots, s_c \rangle$, the column indices of the d largest elements $s_{\ell-d+1}, \dots, s_\ell$ and the size ℓ . We say that θ is a *partial model of order* (c, d) . Obviously, a model of order (c, d) becomes complete if $c + d = \ell$. The idea of the algorithm from [45,46] is to increase c and d in the partial model until we get a good quality complete model.

The total number of partial models of order $(1, 1)$ in the matrix with n columns is $n(n - 1)$. At the first step of the algorithm we select t best partial models of order $(1, 1)$. Next we try to derive partial models of order $(2, 1)$ from the selected partial models of order $(1, 1)$. Pick t best models of order $(2, 1)$. At the step two we try to extend them to partial models of order $(2, 2)$. We continue this process until we get t models of order $(\lceil \ell/2 \rceil, \lceil \ell/2 \rceil)$. Overall complexity of the algorithm is $O(tn^3m)$ [45,46].

5.14. OP-cluster

The order preserving cluster (OP-cluster) model is proposed by Liu and Wang in [47]. This model is similar to the OPSM-model discussed above and can be considered, in some sense, as its generalization. It aims at finding biclusters where the features follow the same order of values in all the samples. However, when two feature values in a sample are close enough, they are considered indistinguishable and allowed to be in any order in the sample. Formally, if features

$i, i + 1, \dots, i + \Delta i$ are ordered in a nondecreasing sequence in a sample j (i.e., $a_{ij} \leq a_{i+1,j} \leq \dots \leq a_{i+\Delta i,j}$) and a user-specified *grouping threshold* $\delta > 0$ is given, the sample j is called *similar* on these attributes if

$$a_{i+\Delta i,j} - a_{ij} < G(\delta, a_{ij}),$$

where G is a grouping function defining where feature values are equivalent. Such a sequence of features are called a *group* for sample j . The feature i is called the *pivot point* of this group. The function G may be defined in different ways. The authors use a simple choice

$$G(\delta, a_{ij}) = \delta a_{ij}.$$

Next, a sequence of features is said to show an *UP pattern* in a sample if it can be partitioned into groups so that the pivot point of each group is not smaller than the preceding value in the sequence. Finally, a bicluster is called an *OP-cluster* if there exists a permutation of its features such that they all show an UP pattern.

The authors presented an algorithm for finding OP-clusters with no less than the required number of samples ns and number of features nf . The algorithm essentially searches through all ordered subsequences of features existing in samples to find maximal common ones, but due to a representation of feature sequences in a tree form allowing for an efficient pruning technique the algorithm is sufficiently fast in practice to apply to real data.

5.15. Supervised classification via maximal δ -valid patterns

In [48] the authors defined a δ -valid pattern as follows. Given a data matrix $A = (a_{ij})_{m \times n}$ and $\delta > 0$, a submatrix $(\mathcal{F}_0, \mathcal{S}_0)$ of A is called a δ -valid pattern if

$$\forall i \in \mathcal{F}_0 \max_{j \in \mathcal{S}_0} a_{ij} - \min_{j \in \mathcal{S}_0} a_{ij} < \delta. \quad (37)$$

The δ -valid pattern is called *maximal* if it is not a submatrix of any larger submatrix of A , which is also a δ -valid pattern. Maximal δ -valid patterns can be found using the SPLASH algorithm [49].

The idea of the algorithm is find an optimal set of δ -patterns such that they cover the samples' set. It can be done using a greedy approach selecting first most statistically significant and most covering patterns. Finally, this set of δ -patterns is used to classify the test samples (samples with unknown classification). For more detailed description of the technique we refer the reader to [48].

5.16. cMonkey

cMonkey is another statistical method for biclustering that has been recently introduced by Reiss et al. [50]. The method is developed specifically for genetic data and works at the same time with gene sequence data, gene expression data (from a microarray) and gene network association data. It constructs one bicluster at a time with an iterative procedure. First, the bicluster is created either randomly or from the result of some other clustering method. Then, on each step, for each sample and feature it is decided whether it should be added to/removed from the bicluster. For this purpose, the probabilities of the presence of the considered sample or feature in the bicluster with respect to the current structure of the bicluster at the three data levels is computed, and a simulated annealing formula is used to make the decision about the update on the basis of the computed probabilities. This way, even when these probabilities are not high, the update has a nonzero chance to occur (that allows escapes from local optima as in any other simulated annealing technique for global optimization). The actual probability of the update also depends on the chosen annealing schedule, so earlier updates have normally higher probability of acceptance while the later steps get almost identical to local optimization. We refer the reader to [50] for the detailed description of the Reiss et al. work.

6. Applications

6.1. Biomedicine

The importance of data analysis in life sciences is steadily increasing. Up to recently, biology was a descriptive science providing relatively small amount of numerical data. However, nowadays it has become one of the main applications of data mining techniques operating on massive data sets. This transformation can be, particularly, attributed to two recent

advances which are complementary to each other. First, the Human Genome Project and some other genome-sequencing undertakings have been successfully accomplished. They have provided the DNA sequences of the human genome and the genomes of a number of other species having various biological characteristics. Second, revolutionary new tools able to monitor quantitative data on the genome-wide scale have appeared. Among them, there are the *DNA microarrays* widely used at the present time. These devices measure gene expression levels of thousands of genes simultaneously, allowing researchers to observe how the genes act in different types of cells and under various conditions. A typical microarray data set includes few classes of samples each of which represents a certain medical condition or type of cells. There may be also a control class (group) representing healthy samples or cells in a predominant state.

Microarray data sets are a very important application for biclustering. When biclustering is performed with high reliability, it is possible not only to diagnose conditions represented by sample clusters, but also identify genes (features) responsible for them or serving as their markers. A great variety of biclustering methods that have been used in microarray data analysis are described in [19,27,29,33,37,39,43,45,46,48].

Among the publicly available microarray data sets that are often used to test biclustering algorithms we should point out:

- ALL vs. AML data set [51,52];
- HuGE (Human GENome) data set [53,54];
- Colon Cancer data set [55];
- B-cell lymphoma data set [56,57];
- Yeast Microarray data set [58,59];
- Lung Cancer data set [60];
- MLL Leukemia data set [61].

Apart from DNA microarray data, biclustering was used in a number of other biomedical applications. In [47] biclustering was applied to drug activity data to associate common properties of chemical compounds with common groups of their descriptors (features). Also [43] presents the application of biclustering to nutritional data. Namely, each sample is associated with a certain food while each feature is an attribute of the food. The goal was to form clusters of foods similar with respect to a subset of attributes.

6.2. Text mining

Another interesting application of biclustering approaches is in text mining. In a classical text representation technique known as the *vector space model* (sometimes also called *bag-of-words model*) we operate with a data matrix $A = (a_{ij})_{m \times n}$, where each row (feature) correspond to a word (or term), each column (sample) to a document and the value of a_{ij} is a certain weight of word i in the document j . In the simplest case, this weight can be, for example, the number of times word i appears in text j .

Text mining techniques are of crucial importance for text indexing, various document organization, text filtering, web search, etc. For a recent detailed survey on text classification techniques we refer the reader to [62].

Classical mining of the text data involve one-way clustering of either word, or document data into classes of related words or documents, respectively [62–64]. Biclustering of text data allows not only to cluster documents and words simultaneously, but also discovers important relations between document and word classes. Successful biclustering approaches for text mining are based on SVD-related [24] or information theoretic techniques [31,32].

Some of the well-known data sets for text mining include:

- *20 Newsgroups data set* (collected by Lang [65], available at [66,67]);
- *SMART collection* [68];
- *Reuters-21578* [69];
- *RCV1-v2/LYRL2004* [70,71].

6.3. Others

Biclustering has been also applied to a number of other areas. Among them is marketing. In collaborative filtering the goal is to find groups of customers with similar attitude or behavior toward a subset of products. The practical value

of this problem is to plan target marketing or recommendation system. Then, if each sample represents a customer, each feature represents a product, and each data value expresses in some way the investigated attitude or behavior, biclustering comes as a handy tool to handle the problem. There is a number of papers considering collaborative filtering of movies, where the data values are either binary (i.e., showing whether a certain customer watched a certain movie or not) or express the rate at which a customer is assigned to a movie. We refer the reader to papers [22,23,72–74] for details.

Finally, biclustering has been also used for dimensionality reduction of databases via automatic subspace clustering of high dimensional data [75], electoral data analysis (finding groups of countries with similar electoral preferences/political attitude toward certain issues) [18], and analyzing foreign exchange data (finding subsets of currencies, whose exchange rates create similar patterns over certain subsets of months) [43].

7. Discussion and concluding remarks

In this survey we reviewed the most widely used and successful biclustering techniques and their related applications. Generally speaking many of the approaches rely on not mathematically strict arguments and there is a lack of methods to justify the quality of the obtained biclusters. Furthermore, additional efforts should be made to connect properties of the biclusters with phenomena relevant to the desired data analysis.

Therefore, future development of biclustering should involve more theoretical studies of biclustering methodology and formalization of its quality criteria. More specifically, as we observed that the biclustering concept has remarkable interplay with algebraic notion of the SVD, we believe that biclustering methodology should be further advanced in the direction of algebraic formalization. This should allow effective utilization of classical algebraic algorithms. In addition, more formal setup for desired class separability can be achieved with establishing new theoretical results similar in spirit to conic separability theorem in [30].

The number of biclustering applications can be also extended with other areas, where simultaneous clustering of data samples and features (attributes) makes a lot of sense. For example, one of the promising directions may be biclustering of stock market data. This way clustering of equities may reveal to us groups of companies whose performance is dependent on the same (but possibly hidden) factors, while clusters of trading days may reveal unknown patterns of stock market returns.

To summarize, one should emphasize that further successful development of biclustering theory and techniques is essential for the progress in data mining and its applications such as text mining, computational biology, etc.

References

- [1] Abello J, Pardalos PM, Resende MG, editors. Handbook of massive data sets. Dordrecht: Kluwer Academic Publishers; 2002.
- [2] Barnes ER, Hoffman AJ, Rothblum UG. Optimal partitions having disjoint convex and conic hulls. *Mathematical Programming* 1992;54: 69–86.
- [3] Granot D, Rothblum UG. The Pareto set of the partition bargaining game. *Games and Economic Behavior* 1991;3:163–82.
- [4] Hwang FK, Onn S, Rothblum UG. Linear shaped partition problems. *Operations Research Letters* 2000;26:159–63.
- [5] Johnson SC. Hierarchical clustering schemes. *Psychometrika* 1967;2:241–54.
- [6] MacQueen JB. Some methods for classification and analysis of multivariate observations. In: *Proceedings of the fifth symposium on mathematics and probability*. CA, USA: Berkeley; 1967.
- [7] Kohonen T. *Self-organization maps*. Berlin-Heidelberg: Springer; 1995.
- [8] Cristianini N, Shawe-Taylor J. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge: Cambridge University Press; 2000.
- [9] Vapnik V. *The nature of statistical learning theory*. Berlin: Springer; 1999.
- [10] Boros E, Hammer P, Ibaraki T, Cogan A. Logical analysis of numerical data. *Mathematical Programming* 1997;79:163–90.
- [11] Boros E, Hammer P, Ibaraki T, Cogan A, Mayoraz E, Muchnik I. An implementation of logical analysis of data. *IEEE Transactions Knowledge and Data Engineering* 2000;12:292–306.
- [12] Xu R, Wunsch D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks* 2005;16:645–8.
- [13] Madeira SC, Oliveira AL. Biclustering algorithms for biological data analysis: a survey. *IEEE Transactions on Computational Biology and Bioinformatics* 2004;1:24–45.
- [14] Tanay A, Sharan R, Shamir R. Biclustering algorithms: a survey, *Handbook of bioinformatics*, 2004, to appear. Available at (http://www.cs.tau.ac.il/~rshamir/papers/bicrev_bioinfo.ps), last accessed August 2006.
- [15] Biclustering—Wikipedia, the Free Encyclopedia, (<http://en.wikipedia.org/wiki/Biclustering>), last accessed August 2006.
- [16] Quertermous Laboratory, Stanford University, HeatMap Builder Software, (<http://quertermous.stanford.edu/heatmap.htm>), last accessed August 2006.

- [17] Golub GH, Van Loan CF. Matrix computations. Baltimore, MD: The Johns Hopkins University Press; 1996.
- [18] Hartigan JA. Direct clustering of a data matrix. *Journal of the American Statistical Association* 1972;67:123–9.
- [19] Cheng Y, Church GM. Biclustering of expression data. In: Proceedings of the eighth international conference on intelligent systems for molecular biology. 2000. p. 93–103.
- [20] Cheng Y, Church GM. Biclustering of expression data. Supplementary information, (<http://arep.med.harvard.edu/biclustering/>), last accessed August 2006.
- [21] Bryan K, Cunningham P, Bolshakova N. Biclustering of expression data using simulated annealing. In: Proceedings of the 18th IEEE symposium on computer-based medical systems. 2005. p. 383–8.
- [22] Yang J, Wang W, Wang H, Yu P. δ -Clusters: capturing subspace correlation in a large data set. In: Proceedings of the 18th IEEE international conference on data engineering. 2002. p. 517–28.
- [23] Yang J, Wang W, Wang H, Yu P. Enhanced biclustering on expression data. In: Proceedings of the third IEEE conference on bioinformatics and bioengineering. 2003. p. 321–7.
- [24] Dhillon IS. Co-clustering documents and words using bipartite spectral graph partitioning. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining(KDD), August 26–29, 2001, San Francisco, CA, USA [also UT CS technical report #TR-01-05, March 2001].
- [25] Kluger Y, Basri R, Chang JT, Gerstein M. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research* 2003; 703–16.
- [26] Bapat RB, Raghavan TES. Non-negative matrices and applications. Cambridge, UK: Cambridge University Press; 1997 [chapter 6].
- [27] Cho H, Dhillon IS, Guan Y, Sra S. Minimum sum-squared residue co-clustering of gene expression data. In: Proceedings of the fourth SIAM international conference on data mining. 2004.
- [28] Cho H, Guan Y, Sra S. Co-clustering Software, Version 1.1 (2005), (<http://www.cs.utexas.edu/users/dml/Software/cocluster.html>).
- [29] Busygin S, Jacobsen G, Krämer E. Double conjugated clustering applied to leukemia microarray data. *SIAM data mining workshop on clustering high dimensional data and its applications*, 2002.
- [30] Busygin S, Prokopyev OA, Pardalos PM. Feature selection for consistent biclustering via fractional 0–1 programming. *Journal of Combinatorial Optimization* 2005;10/1:7–21.
- [31] Dhillon IS, Mallela S, Modha DS. Information-theoretic co-clustering. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining(KDD). August 2003. p. 89–98.
- [32] Banerjee A, Dhillon IS, Ghosh J, Merugu S, Modha DS. Generalized maximum entropy approach to Bregman co-clustering and matrix approximations. In: Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining(KDD), August 2004. p. 509–14.
- [33] Sheng Q, Moreau Y, De Moor B. Biclustering microarray data by Gibbs sampling. *Bioinformatics* 2003;19:ii196–205.
- [34] Casella G, George EI. Explaining the Gibbs sampler. *The American Statistician* 1992;46:167–74.
- [35] Tanay A. Computational analysis of transcriptional programs: function and evolution. PhD thesis, August 2005. Available at (http://www.cs.tau.ac.il/~rshamir/theses/amos_phd.pdf), last accessed August 2006.
- [36] Tanay A, Sharan R, Shamir R. Discovering statistically significant biclusters in gene expression data. *Bioinformatics* 2002;18:S136–44.
- [37] Tanay A, Sharan R, Kupiec M, Shamir R. Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proceeding of the National Academy of Science USA* 2004;101:2981–6.
- [38] Computational Genomics Laboratory, School of Computer Science, Tel Aviv University, Israel, EXPANDER, A gene expression analysis and visualization software, (<http://www.cs.tau.ac.il/~rshamir/expander/expander.html>), last accessed August 2006.
- [39] Getz G, Levine E, Domany E. Coupled two-way clustering analysis of gene microarray data. *PNAS* 2000;97:12079–84.
- [40] Blatt M, Wiseman S, Domany E. Data clustering using a model granular magnet. *Neural Computation* 1997;9:1805–42.
- [41] Domany E. Super-paramagnetic clustering of data. *Physica A* 1999;263:158–69.
- [42] Weizmann Institute of Science, The coupled two way clustering algorithm, (<http://ctwc.weizmann.ac.il/>), last accessed August 2006.
- [43] Lazzeroni L, Owen A. Plaid models for gene expression data. *Statistica Sinica* 2002;12:61–86.
- [44] Plaid models, for microarrays and DNA expression, (<http://www-stat.stanford.edu/~owen/plaid/>), last accessed August 2006.
- [45] Ben-Dor A, Chor B, Karp R, Yakhini Z. Discovering local structure in gene expression data: the order-preserving submatrix problem. In: Proceedings of the sixth annual international conference on computational biology (RECOMB '02). New York: ACM Press; 2002. p. 49–57.
- [46] Ben-Dor A, Chor B, Karp R, Yakhini Z. Discovering local structure in gene expression data: the order-preserving submatrix problem. *Journal of Computational Biology* 2003;10:373–84.
- [47] Liu J, Wang W. OP-cluster: clustering by tendency in high dimensional space. In: Proceedings of the third IEEE international conference on data mining. 2003. p. 187–94.
- [48] Califano A, Stolovitzky S, Tu Y. Analysis of gene expression microarrays for phenotype classification. In: Proceedings of the eighth symposium on intelligent systems for molecular biology, San Diego, 2000.
- [49] Califano A, SPLASH: structural pattern localization analysis by sequential histograms. *Bioinformatics* 16 (2000), pp. 341–57. The algorithm is available at (<http://www.research.ibm.com/splash/>).
- [50] Reiss DJ, Baliga NS, Bonneau R. Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. *BMC Bioinformatics* 2006;7:280.
- [51] Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP. et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 1999;286:531–7.
- [52] Cancer Program Data Sets, BROAD Institute, MIT, (http://www.broad.mit.edu/cgi-bin/cancer/data_sets.cgi), last accessed August 2006.
- [53] Hsiao L-L, Dangond F, Yoshida T, Hong R, Jensen RV, Misra J. et al. A compendium of gene expression in normal human tissues. *Physiological Genomics* 2001;7:97–104.

- [54] The Human Gene Expression Index, (<http://www.hugeindex.org>), last accessed August 2006.
- [55] Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences* 1999; (96) 6745–50.
- [56] Alizadeh AA, Eisen MB, Davis RE, Ma C, Lossos IS, Rosenwald A. et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature* 2000;403:503–11.
- [57] Lymphoma/Leukemia Molecular Profiling Project, NIH, (<http://lmpp.nih.gov/lymphoma/index.shtml>), last accessed August 2006.
- [58] Tavazoie S, Hughes JD, Campbell MJ, Cho RJ, Church GM. Systematic determination of genetic network architecture. *Nature Genetics* 1999;22:281–5.
- [59] Systematic determination of genetic network architecture, Harvard University, (http://arep.med.harvard.edu/network_discovery/), last accessed August 2006.
- [60] Gordon GJ, Jensen RV, Hsiao L-L, Gullans SR, Blumenstock JE, Ramaswamy S. et al. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Research* 2002;62:4963–7.
- [61] Armstrong SA, Staunton JE, Silverman LB, Pieters R, den Boer ML, Minden MD. et al. MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics* 2002;30:41–7.
- [62] Sebastiani F. Machine learning in automated text categorization. *ACM Computing Surveys* 2002;34:1–47.
- [63] Baker LD, McCallum AK, Distributional clustering of words for text classification In : Croft WB, Moffat A, van Rijsbergen CJ, Wilkinson R, Zobel J, editors. *Proceedings of SIGIR-98, 21st ACM international conference on research and development in information retrieval*. New York, ACM Press; US: 1998 p. 96–103.
- [64] Crouch CJ. A cluster-based approach to thesaurus construction. In: *Proceedings of the 11th annual international ACM SIGIR conference on research and development in information retrieval*. France: Grenoble; 1988. p. 309–20.
- [65] Lang K. NewsWeeder: learning to filter netnews. In: *Proceedings of the 12th international conference on machine learning*. San Mateo, CA, USA: Morgan Kaufmann Publishers Inc.; 1995. p. 331–9.
- [66] CMU Text Learning Group Data Archives, 20 Newshroup DataSet, (<http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html>), last accessed August 2006.
- [67] 20 Newsgroups Data Set, (<http://people.csail.mit.edu/jrennie/20Newsgroups/>), last accessed August 2006.
- [68] Cornell University, SMART, (<ftp://ftp.cs.cornell.edu/pub/smart>), last accessed August 2006.
- [69] Lewis DD. Reuters-21578 text categorization test collection, Distribution 1.0, 2004. (<http://www.daviddlewis.com/resources/testcollections/reuters21578/>), last accessed August 2006.
- [70] Lewis DD. RCV1-v2/LYRL2004: The LYRL2004 distribution of the RCV1-v2 text categorization test collection (14 October 2005 Version). (http://www.jmlr.org/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm), last accessed August 2006.
- [71] Lewis DD, Yang Y, Rose T, Li F. RCV1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research* 2004;5:361–97.
- [72] Ungar L, Foster DP. A formal statistical approach to collaborative filtering. In: *Proceedings of the conference on automated learning and discovery (CONALD'98)*. 1998.
- [73] Hofmann T, Puzicha J. Latent class models for collaborative filtering. In: *Proceedings of the 16th international joint conference on artificial intelligence*. 1999. p. 688–93.
- [74] Wang H, Wang W, Yang J, Yu P. Clustering by pattern similarity in large data sets. In: *Proceedings of the 2002 ACM SIGMOD international conference on management of data*. 2002. p. 394–405.
- [75] Agrawal R, Gehrke J, Gunopulos D, Raghavan P. Automatic subspace clustering of high dimensional data for data mining applications. In: *Proceedings of the 1998 ACM SIGMOD international conference on management of data*. 1998. p. 94–105.
- [76] Pardalos PM, Busygin S, Prokopyev OA. On biclustering with feature selection for microarray data sets. In: Mondaini R, editor. *BIOMAT 2005—international symposium on mathematical and computational biology*. Singapore: World Scientific; 2006. p. 367–78.