Co-clustering: A Survey of the Main Methods, Recent Trends, and Open Problems

ELENA BATTAGLIA, Computer Science, Università degli Studi di Torino, Turin, Italy
FEDERICO PEIRETTI, Computer Science, Università degli Studi di Torino, Turin Italy
RUGGERO GAETANO PENSA, Computer Science, Università degli Studi di Torino, Turin Italy

Since its early formulations, co-clustering has gained popularity and interest both within and outside the machine learning community as a powerful learning paradigm for clustering high-dimensional data with good explainability properties. The simultaneous partitioning of all the modes of the input data tensors (rows and columns in a data matrix) is both a method for improving clustering on one mode while performing dimensionality reduction on the other mode(s), and a tool for providing an actionable interpretation of the clusters in the main mode as summaries of the features in each other mode(s). Hence, it is useful in many complex decision systems and data science applications. In this article, we survey the the co-clustering literature by reviewing the main co-clustering methods, with a special focus on the work done in the past 25 years. We identify, describe, and compare the main algorithmic categories and provide a practical characterization with respect to similar unsupervised techniques. Additionally, we try to explain why it is still a powerful tool despite the apparent recent decreasing interest shown by the machine learning community. To this purpose, we review the most recent trends in co-clustering research and outline the open problems and promising future research perspectives.

CCS Concepts: • **Computing methodologies → Cluster analysis** • **General and reference → Surveys and overviews** • **Information systems → Clustering**;

Additional Key Words and Phrases: Co-clustering, high-dimensional data clustering, review

ACM Reference Format:
Elena Battaglia, Federico Peiretti, and Ruggero Gaetano Pensa. 2024. Co-clustering: A Survey of the Main Methods, Recent Trends, and Open Problems. *ACM Comput. Surv.* 57, 2, Article 48 (November 2024), 33 pages.
https://doi.org/10.1145/3698875

1 Introduction

Finding homogeneous clusters of objects is an important task in data analysis, with applications in a wide range of fields, such as text analysis [20], bioinformatics [105], e-commerce [38], astronomy [64], and psychology [24]. Several clustering techniques and algorithms have been developed, and their applications have given effective results in many scenarios. However, when the data are high-dimensional and/or very sparse, the traditional methods struggle in finding meaningful clustering solutions due to the well-known problem of the curse of dimensionality. Solutions to this

Authors' Contact Information: Elena Battaglia, Computer Science, Università degli Studi di Torino, Turin, Italy; e-mail: elenabatt.eb@gmail.com; Federico Peiretti, Computer Science, Università degli Studi di Torino, Turin, Italy; e-mail: federico.peiretti@unito.it; Ruggero Gaetano Pensa, Computer Science, Università degli Studi di Torino, Turin, Italy; e-mail: ruggero.pensa@unito.it.

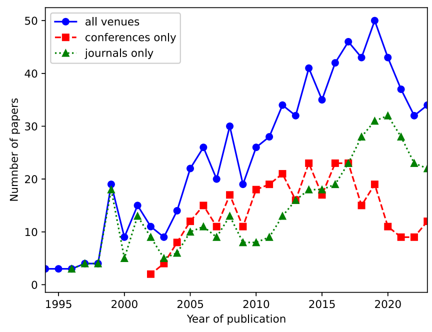
ACM 0360-0300/2024/11-ART48
https://doi.org/10.1145/3698875

problem range from applying dimensionality reduction or matrix factorization approaches prior to clustering [174], to searching clusters in subspaces [1]. Co-clustering, among others, has gained popularity in the past 30 years: It consists in the simultaneous partitioning of the rows and columns of a data matrix [53]. Due to its intrinsic capability of exploiting the latent relationships between objects and their own attributes, it enables the discovery of coherent clusters of similar rows and their interplay with corresponding attribute clusters. In a nutshell, it applies clustering in one mode (e.g., the rows) of a matrix while performing dimensionality reduction on the other mode (e.g., the columns) and vice versa. When applied to high-dimensional data, not only does co-clustering improve the clustering performances, but it also increases the interpretability of the results, showing which attributes or groups of attributes are the most important for each cluster of objects. More interestingly, such interpretation is not obtained *ex post*, but drives directly the partitioning of the matrix, thus making co-clustering among the few intrinsically explainable clustering methods for high-dimensional data.

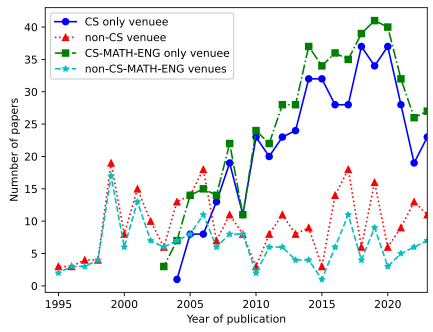
Although some early formulations were proposed in 1972 with the name of direct clustering [78] and in 1980 with the name of simultaneous clustering [23], it is in the mid-'90s that co-clustering started to be studied and developed more intensively, also under different names such as: simultaneous clustering [70], biclustering [42], two-mode partitioning [143] or clustering [118], block clustering [71], and double k-means [117]. The term "co-clustering" was first used by Dhillon et al. in 2001 [53] and, since then, it has been almost universally adopted to cover all similar methods. There exist many co-clustering techniques, tailored for different data types and application scenarios [74, 116, 137]. Among the others, co-clustering has proven its effectiveness when applied on gene expression data—i.e., matrices reporting the level of expression of genes in different biological conditions/samples—and contingency tables—i.e., matrices reporting the co-occurrences between two sets of categorical values, such as documents/words, customers/products or users/movies data objects. In particular, the latter are typically high-dimensional and extremely sparse and, thus, they cannot be handled by classical clustering methods. Furthermore, the columns of these matrices represent instances of the same categorical feature (words, products, or movies, in the examples above), thus grouping them in clusters makes sense and provides additional useful information. Gene expression data matrices, instead, although can be also viewed as a special case of contingency tables, have different mathematical properties. They, in fact, are typically dense and, in most cases, their entries are not related to co-occurrences (frequencies or counts), but consist of generic real (or sometimes integer) numbers.

Many extensions of the traditional co-clustering problem have been studied: For example, there are algorithms to perform co-clustering in a distributed-scenario [127], constrained, and semi-supervised methods [107, 134, 139, 153], and algorithms for co-clustering heterogeneous data [67, 88]. Tensor co-clustering can be also viewed as an extension of 2-way co-clustering. Just as relations between two sets of categorical values can be expressed in a contingency table, so relations among three or more sets can be represented via tensors (or n-way data). These contingency tensors are usually even more complex to cluster than matrices, and tensor co-clustering (i.e., simultaneous partitioning of all the modes of a tensor) becomes a fundamental tool to give a compact representation of the embedded clusters [80, 130].

Despite the good properties shown by co-clustering, it seems that the research interest of the scientific community for this topic is decreasing in the past five years, as shown in Figure 1, where we plot the number of papers mentioning "co-clustering" (or closely related terms, such as "coclustering," "co-cluster," "simultaneous clustering," or "two-mode clustering") in the title, according to the bibliographic database Scopus. However, if we consider the number of papers citing articles mentioning the same terms (see Figure 2), then the trend is still increasing, meaning that co-clustering is till conveying significant scientific interest in the research communities.

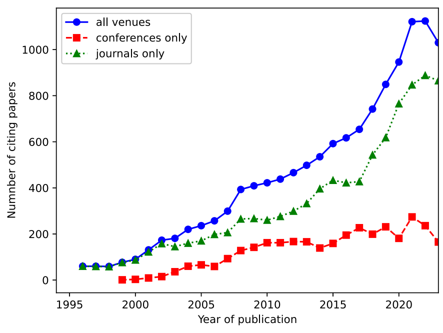


(a) Papers published yearly for different types of venue.

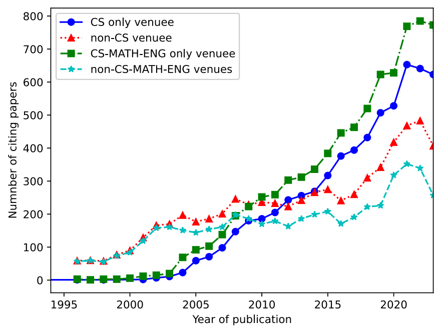


(b) Papers published yearly for different research fields.

Fig. 1. Number of research articles published yearly and mentioning co-clustering-related keywords in their title (source: Scopus). In the left plot, the number of articles is presented for different types of venues (journal articles and conference proceeding papers). In the right plot, the number of articles is shown according to the subject area (computer science only, computer science with mathematics and engineering, non–computer science venues, and venues other than computer science, mathematics, and engineering).



(a) Papers published yearly for different types of venue.



(b) Papers published yearly for different research fields.

Fig. 2. Number of research articles published yearly and citing papers mentioning co-clustering-related keywords in their title (source: Scopus). In the left plot, the number of articles is presented for different types of venues (journal articles and conference proceeding papers). In the right plot, the number of articles is shown according to the subject area (computer science only, computer science with mathematics and engineering, non–computer science venues, and venues other than computer science, mathematics, and engineering).

In this article, we survey the scientific contributions in the field of co-clustering, with a special emphasis on those following the work done by Dhillon et al. in 2001 [53]. We present the different categories of algorithms proposed so far from a theoretical and application point of view. Furthermore, we analyze the main reasons for the apparent decline in the interest for proposing new approaches or extensions of existing ones. In doing so, we also point out scarcely addressed issues and promising research directions. Rather than briefly enumerating and describing the different state-of-the-art methods, we provide an in-depth presentation of the theoretical foundations of the different categories of approaches. We discuss the main methods for each of them, also surveying their extensions designed for multi-view and n-way data. So, the utility of our survey is twofold: It can be used by the inexperienced researchers to learn when to use co-clustering for their data science applications and which class of methods is best suited for them; however, the expert readers could find in our survey a guide for driving their future research work on the topic or to gain more insights on some specific methods. In both cases, to the best of our knowledge, this is the first comprehensive review specifically dedicated to general co-clustering algorithms.

Our article is organized as follows: In Section 2, we briefly present some closely related surveys in similar domains. We report the formal definition of co-clustering and what distinguishes it from other similar approaches in Section 3. In Section 4, we review the main categories of approaches for matrix co-clustering. A theoretical comparison of the different categories of methods is discussed in Section 5. Tensor and multi-view co-clustering methods are presented in Section 6. In Section 7, we discuss the most recent research trends, the open problems, and some future research directions. In Section 8, we draw some conclusions.

2 Related Work

Clustering is almost as old as artificial intelligence and, consequently, its related algorithms have been extensively studied in many reviews by the computer science community since the ’80s. Most surveys focus on specific algorithmic aspects of data clustering or on clustering of particular data types. One of the first books addressing many general aspects of clustering algorithms is the one by Jain and Dubes [91]. Ten years later, Jain published an extensive review on clustering [92], while the most recent relevant example of general review on clustering algorithms and their applications is Reference [173]. As said before, other more recent reviews address more specific topics, such as fuzzy clustering [16, 17], clustering of time series [5, 111] or data streams [51], clustering for high-dimensional data [100, 151], or special algorithmic solutions [63, 83]. Among the last published surveys, Reference [12] focuses on interactive clustering techniques, Reference [10] addresses spatiotemporal data clustering, References [49, 89, 188] survey parallel and efficient clustering solutions for data streams and big data in general, and Reference [31] focuses on semi-supervised clustering.

Despite this great interest for clustering, very few papers are specifically devoted to surveying co-clustering approaches. The first notable example is the 2004 survey on biclustering by Madeira and Oliveira [116], where the authors focus on co-clustering and biclustering solutions for gene expression data (and biological data in general). However, as we will point out in Section 3, biclustering has completely different goals (and methods) as compared to co-clustering. Nonetheless, the review also considers and describes some of the earlier co-clustering approaches. More recently, Madeira has contributed to the field with other surveys addressing the problem of the search of coherent triclusters (extension of biclusters in higher-order data) [80] and the impact of metrics in identifying biclusters [124]. Co-clustering is not the focus of these reviews, but a few approaches are mentioned as well.

A recent survey paper more specifically tailored on co-clustering algorithms is the one by Lin et al. [112], which, however, only focuses on approaches based on matrix factorization. In their paper, the authors present six co-clustering algorithms and conduct experiments by applying them on some real-world datasets. The goal of this overview is to present the strengths and limitations of existing approaches based mainly on non-negative matrix factorization. Another more recent survey is fully dedicated to a detailed and in-depth analysis of the latent block model for co-clustering [22]. Our survey, instead, also considers other categories of approaches.

Another important work specifically addressing co-clustering methods is the 2013 book authored by Govaert and Nadif [74]. It presents the theoretical and algorithmic details of many different approaches, focusing on model-based algorithms in particular. Other approaches, including information theoretic methods, are also considered, but, for instance, algorithms based on matrix-factorization are not extensively analyzed. Nonetheless, the book also presents some experimental results showing the comparative performances of different approaches. Since the book is not recent, it does not include relevant recent work on co-clustering or significant extensions to higher-order data. It must be said that it is not a survey but a textbook presenting some popular and less popular algorithms on co-clustering. As such, it also includes an introductory chapter on standard clustering.

When the broader problem of clustering high-dimensional data is considered, subspace clustering is a method that shares some similar characteristics with co-clustering, as we will point out in Section 3. However, the most valuable surveys that explore this specific research field [100, 126, 131, 151] only sporadically mention co-clustering algorithms as examples of related methods. It is worth considering that subspace clustering has more similarities with biclustering than co-clustering, as clearly specified in Reference [124]. Differently from all the works mentioned above, we specifically focus on co-clustering. Nonetheless, we stress the differences between it and subspace clustering to help the reader dispel any doubts about this matter.

3 Problem Formulation

In this section, we formally define the problem of co-clustering. However, before entering the details of the definitions, we introduce the notation needed to unify the formal presentation of the main categories of co-clustering approaches. Since a matrix is a special case of tensors with only two modes (the rows and the columns), our notation will refer to the problem of co-clustering for a generic n-way tensor.

3.1 Notation

Data tensors are denoted by Euler script letters, e.g., $\mathcal{X}$ or $\mathcal{T}$. The generic entry of a tensor $\mathcal{X}$ is denoted by $x_{i_1 \dots i_N}$, where N is the number of modes of tensor $\mathcal{X}$ and, for $j = 1, \dots, n$, i_j is the ith element on the jth mode. Thus, denoted by I_j the dimension of the tensor on the jth mode, the tensor can also be written as

$$\mathcal{X} = (x_{i_1 \dots i_N})_{\substack{i_j = 1, \dots, I_j \\ j = 1, \dots, N}}.$$

Notice that a *data matrix* can be considered as a two-way tensor. As a consequence, the notation used for matrices is the same used for tensors. For instance, let $\mathcal{X}$ be a matrix of dimension (n, m); its (i, j)th element is denoted by x_{ij}, where $i = 1, \dots, n, j = 1, \dots, m$ and the matrix will be also denoted as

$$\mathcal{X} = (x_{ij})_{\substack{i = 1, \dots, n \\ j = 1, \dots, m}}.$$

Additional matrices used by the co-clustering algorithms for their computation (other than the input matrix) are denoted by capital letters, e.g., W and D.

Given a set $\mathcal{I}$ with n elements, *the set of all the possible partitions* of the n elements is denoted by $\mathbb{P}_n$. A *partition* (or *clustering*) is denoted by a calligraphic uppercase letter, e.g., $\mathcal{P}, \mathcal{R}$. Since a partition $\mathcal{P} \in \mathbb{P}_n$ is a collection of a certain number k of sets (or *clusters*), we will also write $\mathcal{P} = (\mathcal{P}_1, \dots, \mathcal{P}_k)$. To indicate that the ith element of $\mathcal{I}$ belongs to the hth cluster of $\mathcal{P}$, we will write that $i \in \mathcal{P}_h$ or that $\mathcal{P}(i) = h$. Thus, with abuse of notation, we will use the same symbol $\mathcal{P}$ to denote both the partition and the function $\mathcal{I} \longrightarrow \mathcal{P}$ that assigns each element of $\mathcal{I}$ to the cluster to which the element belongs.

3.2 Definition of Co-clustering

Before giving a rigorous definition of co-clustering, it is necessary to make a brief digression regarding the terms used to identify this problem. The problem of simultaneously clustering rows and columns is usually called by different names by different authors: *co-clustering*, *biclustering*, *simultaneous clustering*, or *subspace clustering*. Sometimes all or part of these names are used to indicate the same problem, while other authors use different names to indicate slightly different problems. Although there is no agreement between the authors, in this manuscript, we will make the following distinction:

- **Co-clustering** is the simultaneous clustering of the rows and the columns of a data matrix. The idea behind co-clustering is to exploit the duality between objects and features to identify better partitions on both the dimensions of the matrix. It is especially effective when applied to high-dimensional and sparse contingency tables (or co-occurrences tables) to solve problems such as document clustering or customer segmentation.
- **Biclustering** consists in the identification of one (or more) subset of entries in the data matrix that exhibits a coherent behavior. It is principally used in bioinformatics, on gene expression data, especially for discovering functionally related gene sets under different subsets of experimental conditions. The output of this kind of algorithm is a subset of the rows of the data matrix (for instance, genes) and a subset of columns (conditions) according to which the selected rows show a similar behavior (and/or vice versa). For a review of the main biclustering approaches, see References [116, 136, 170].
- **Subspace Clustering** is based on the idea that the rows of the data matrix belong to different subspaces, and the task is to identify these subspaces and to cluster the rows according to the subspace they belong to. The output of such algorithms is a partition of the rows and a basis of the subspaces the clusters belong to. The main methods and concepts for subspace clustering are reviewed in References [1, 101, 131, 160].

The three problems just defined are highly related and sometimes their definitions coincide. For example, in the co-clustering problem, usually the partitions on rows and columns are asked to be exhaustive and exclusive (i.e., each row/column belongs to one and only one cluster); however, if we relax these conditions and we look for non-exhaustive clustering solutions on rows and columns, then the formulation of co-clustering is similar to the one of biclustering. However, also biclustering and subspace clustering are often used as synonyms. The distinction between the two is mainly in the problem formulation: Subspace clustering focus is on the space from which the data are extracted, and the goal is to give a geometrical/algebraic description of the space as union of subspaces. Instead, biclustering methods are thought and used to identify submatrices inside the data matrix with constant or coherent values.

In this manuscript, we will focus only on the first problem, co-clustering, in its stronger definition: simultaneous *hard* clustering of both the dimensions of a data matrix.[1] For a formal definition, let us denote with $\mathbb{P}_n$ the space of all the possible partitions of n elements. A partition $\mathcal{P} \in \mathbb{P}_n$ is a collection of k sets $\mathcal{P}_1, \dots, \mathcal{P}_k$ (also called clusters) such that

$$\cup_{i=1}^{k} \mathcal{P}_i = \mathcal{I}_n; \tag{1}$$

$$\mathcal{P}_i \cap \mathcal{P}_j = \emptyset, \text{ for } i, j = 1, \dots, k, i \neq j \tag{2}$$

$$\mathcal{P}_i \neq \emptyset, \text{ for each } i = 1, \dots, k, \tag{3}$$

where $\mathcal{I}_n$ is a generic set with n elements. A co-clustering of a matrix $\mathcal{A}$ with n rows and m columns is a pair of partitions $(\mathcal{R}, \mathcal{C})$, where $\mathcal{R} \in \mathbb{P}_n$ and $\mathcal{C} \in \mathbb{P}_m$.

Although the requirement of exhaustive and exclusive clusters (conditions (1) and (2)) might seem too strict and not easy to apply in real-word scenarios, there are many applications in which it can be used with effective results, as we will show in the next section. We now present a

[1]For the sake of simplicity, we introduce the co-clustering problem on matrices first. Tensor co-clustering will be addressed in Section 6.

motivational example to explain the rationale behind the definition of co-clustering and why it can be a useful tool for high-dimensional data analysis.

Example 3.1. The purchase habits of the customers of a shop can be expressed by a co-occurrence matrix $\mathcal{A}$, whose rows represent the customers and whose columns are the products sold in the shop. Each entry a_{uv} is the number of items of a particular product v bought by a specific customer u. Imagine that two customers, i and j, have the following purchase vectors:

$$a_i = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$
$$a_j = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The two vectors are not similar at all: There is only one product bought by both customers, i.e., the second one. Now, let us suppose that the first seven columns of the matrix represent different books, while the last three columns represent two video-games. If we group together all the books and all the video-games and we represent the purchases of i and j in this new less granular feature space, then we obtain

$$a_i = \begin{pmatrix} 4 & 0 \end{pmatrix}$$
$$a_j = \begin{pmatrix} 4 & 0 \end{pmatrix}.$$

Hence, in this new representation, it is clear that the two customers have very similar purchase habits (they both buy many books and no video-games) and they should be assigned to the same group when performing clustering. This fact was impossible to understand without having an external knowledge about the meaning of the features (or columns of the matrix).

The example above is intended to show that knowing a high-quality partition on one mode of a data matrix helps improve the quality of the partition also on the other dimension. Thus, looking for both row and column partitions simultaneously should lead to higher-quality partitions on both the dimensions.

From Example 3.1 also emerges another important property of co-clustering, i.e., its intrinsic explainability. The advantage of co-clustering in this sense is twofold: (i) It provides a representation of the data in a low-dimensional space, and (ii) the particular dimensionality reduction operated by co-clustering (where all similar features are grouped together) is usually easy to interpret. In fact:

— The dimensionality reduction makes the clustering results easier to read. For instance, suppose to have a set of points in a 100-dimensional space and to run k-means clustering to partition them in few clusters. Since the centroids identified by the k-means are points in a 100-dimensional space, they cannot be plotted and it is difficult to understand their meaning. On the contrary, if they are represented in a l-dimensional space, with l small enough, then they can be plotted (if $l = 2$ or 3) or at least analyzed more easily.
— The partition on the rows can be used to explain the partition on the columns and vice versa. For instance, following Example 3.1, the presence of two customers in the same cluster can be explained observing that they buy with approximately the same frequency products belonging to the same groups. Conversely, two products are grouped together when bought by the same "kind" of customers. In other words, co-clustering gives as output a clustering of the objects and an explanation of the obtained clusters in the form of a clustering of the features. At the end of a co-clustering algorithm, the large data matrix can be summarized by a reduced number of blocks of data: Such a representation can be obtained by rearranging the rows and columns according to their clustering assignment (see, as an example, Figure 1 in the Supplemental Material). This compact representation

permits to understand, at a glance, the relationships between row clusters and column clusters and facilitates any further analysis.

3.3 Different Data Types and Applications

Co-clustering has successfully proven its efficacy in different applications, such as text mining [41, 53, 54], web mining [37, 110], recommendation systems [60, 68], gene expression analysis [42, 44], graph mining [35], image segmentation [96, 137]. It can represent a task itself or be used as a preliminary step to solve other tasks such as anomaly detection [128, 177], classification of out-of-domain documents [50], transfer learning [165, 181], object detection, and scene categorization [32, 96].

The input of a co-clustering algorithm is a data matrix or tensor; in principle, any kind of data collection that can be represented as n-way tensor can be analyzed using a co-clustering algorithm: textual data arranged in document/term matrices, (temporal) gene expression data, adjacency matrices of graphs, geo-referenced time series, images, and so on. However, different algorithms have been tailored for different data types. There exist co-clustering algorithms for ordinal data [46, 90], temporal and directional data [146, 187], and even functional data [152] and mixed type data [149]. Anyway, the vast majority of the co-clustering methods in literature are thought to handle continuous data or binary/categorical data arranged in co-occurrence data tables. The first group includes, among others, gene expression data and images. As pointed out before, for this kind of data the conditions of co-clustering (i.e., exhaustive and exclusive partitions on both dimensions) are often too strict and biclustering or triclustering algorithms allowing the discovery of overlapping and non-exhaustive clusters are preferred [80, 116]. However, some co-clustering algorithms for continuous data do exist [44, 78, 97, 117, 119].

Most co-clustering algorithms are explicitly designed to work on binary and discrete tensors such as contingency tables (also called co-occurrences tables). Some examples are document/term and customer/product matrices or user/movie/tag tensors. Although these algorithms are usually applied to contingency tables, the only condition required is the input to be positive real-valued, thus, in theory, they can also work on other types of data (such as images or gene expression matrices). However, this class of co-clustering algorithms expresses its full potential on sparse data matrices or tensors and hence its usage is not recommended on that kind of data.

It is important to notice that not all matrix methods can be adapted to work on tensors. In fact, even though the general problem of co-clustering can be naturally formulated on tensors, every method has its own specificity, such as the optimization function and/or partition update strategy, and they cannot be straightforwardly extended to their hypothetical n-way formulation. Another reason limiting the direct reformulation of matrix methods to work on tensors is that they would be computationally too expensive when applied on n-ary ($n > 2$) data. Consequently, in this article, we address the two problems separately, and we start by presenting, in the following section, the main co-clustering approaches for (sparse) positive valued data matrices, whose primary application is on contingency tables, as explained above.

4 Co-clustering Approaches

The main methods for contingency matrix co-clustering can be roughly divided into four groups: spectral methods [53, 97], which transform the co-clustering problem into a partitioning problem on a bipartite graph; matrix factorization–based methods, such as Block Value Decomposition [115] and Non-Negative Matrix Tri-Factorization [56]; model-based or probabilistic approaches [7, 73], which assume the data to be generated by a probabilistic model and try to recover the underlying model parameters; methods that formulate the co-clustering problem as an optimization problem for some measure of the quality of the co-clustering solution [6, 14, 54, 141].

Finally, a few methods relying on different paradigms, such as a deep learning model for co-clustering [171], have been recently proposed.

4.1 Spectral Co-clustering

Spectral Co-clustering [53] was proposed by Dhillon in 2001 as a method to simultaneously cluster documents and terms, but it can be used to co-cluster any contingency table. It is an extension of the normalized cut spectral clustering [161], a class of algorithms that compute the eigenvalues of the adjacency matrix of the data similarity graph to perform clustering on fewer dimensions.

A good partition of a graph is one where the edges within each group have high weight, while the edges between different groups have low weights. Hence, a method to partition a graph is to find the node partition $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_k)$ that minimizes the Normalized cut function, defined as

$$Ncut(\mathcal{P}) = \sum_{c=1}^{k} \frac{W(\mathcal{P}_c, \bar{\mathcal{P}_c})}{weight(\mathcal{P}_c)},$$

where $W(\mathcal{P}_c, \bar{\mathcal{P}_c})$ is the sum of the weights of all the edges between a node belonging to $\mathcal{P}_c$ and a node outside $\mathcal{P}_c$, and $weight(\mathcal{P}_c)$ is the sum of the weights of all the edges having at least one node inside $\mathcal{P}_c$. It turns out that solving the minimum normalized cut problem is equivalent to finding the cluster assignment matrix $H = (h_{ic})$, where $h_{ic} = \frac{1}{\sqrt{weight(\mathcal{P}_c)}}$ if the ith node belongs to cluster $\mathcal{P}_c$ and $h_{ic} = 0$ otherwise, which minimizes

$$\min_H \operatorname{tr}(H^T L H) \text{ subject to } H^T D H = I, \tag{4}$$

where L is the Laplacian matrix of the graph. Denoting with D the degree matrix of the graph and with W the weights' matrix, the Laplacian of a graph is defined as

$$L = D - W. \tag{5}$$

Finally, it can be proved that, if the conditions about H are relaxed and H can be any real valued matrix, then the solution of Equation (4) is the matrix U composed by the first k eigenvectors corresponding to the k smallest eigenvalues λ of the generalized eigenvalue problem

$$Lu = \lambda Du. \tag{6}$$

Because of the relaxation of the problem, the resulting matrix U is not a cluster incidence matrix anymore. Thus, the cluster assignment is obtained by applying the k-means clustering algorithm over matrix U.

All the reasoning above can be used to co-cluster a contingency table as follows: Let $\mathcal{A}$ be a contingency table with n rows (representing, for instance, documents) and m columns (for instance, words). It is possible to represent the same data as a bipartite graph, where rows and columns correspond to nodes and an edge between a row i and a column j exists if $a_{ij} > 0$. The weight of the edge between row i and column j is a_{ij}. Then, a clustering of this graph can be obtained with the procedure just described, i.e., the problem is reduced to solving the generalized eigenvalue problem $Lu = \lambda Du$. From the graph partition it is possible to obtain a co-clustering, but this method requires the construction of a Laplacian matrix of shape $(n + m) \times (n + m)$. Instead, it is possible to exploit the bipartite structure of the graph and obtain an algorithm that works directly on matrix $\mathcal{A}$. In fact, letting D_R and W_R be, respectively, the degree matrix and the weights matrix relative to the rows only and D_C and W_C those relative to the columns only, the Laplacian of the bipartite graph can be written as

$$L = D - W = \begin{bmatrix} D_R & 0 \\ 0 & D_C \end{bmatrix} - \begin{bmatrix} 0 & \mathcal{A} \\ \mathcal{A}^T & 0 \end{bmatrix} = \begin{bmatrix} D_R & -\mathcal{A} \\ -\mathcal{A}^T & D_C \end{bmatrix}$$

and the generalized eigenvalue problem (6) can be re-written as

$$\begin{bmatrix} D_R & -\mathcal{A} \\ -\mathcal{A}^T & D_C \end{bmatrix} \begin{bmatrix} u_R \\ u_C \end{bmatrix} = \lambda \begin{bmatrix} D_R & 0 \\ 0 & D_C \end{bmatrix} \begin{bmatrix} u_R \\ u_C \end{bmatrix}.$$

Under the nonsingularity assumption of D_R and D_C, the above equations can be rewritten as

$$D_R^{\frac{1}{2}} u_R - D_R^{-\frac{1}{2}} \mathcal{A} u_C = \lambda D_R^{\frac{1}{2}} u_R$$
$$-D_C^{-\frac{1}{2}} \mathcal{A}^T u_R + D_C^{\frac{1}{2}} u_C = \lambda D_C^{\frac{1}{2}} u_C,$$

and with few additional algebraic manipulations, it can be shown that solving the problem above is equivalent to solve

$$\begin{cases} Bu = \sigma v \\ B^T v = \sigma u \end{cases}, \tag{7}$$

where $B = D_R^{-\frac{1}{2}} \mathcal{A} D_C^{-\frac{1}{2}}$, $u = D_R^{\frac{1}{2}} u_R$, $v = D_C^{\frac{1}{2}} u_C$, and $\sigma = 1 - \lambda$. Equations (7) are precisely the equations that describe the Singular Value Decomposition of matrix B. Thus, a spectral co-clustering of matrix $\mathcal{A}$ can be obtained as follows:

(1) Compute matrix $B = D_R^{-\frac{1}{2}} \mathcal{A} D_C^{-\frac{1}{2}}$;
(2) Compute the singular value decomposition $B = U \Sigma V^T$;
(3) Create matrix $Z = \begin{bmatrix} D_R^{\frac{1}{2}} U \\ D_C^{\frac{1}{2}} V \end{bmatrix}$;
(4) Apply k-means on Z to obtain k clusters.

The final output of the algorithm is a set of k clusters, and each one of them contains both rows and columns. By selecting only the nodes relative to rows (respectively, columns) in each cluster, we obtain the row (respectively, column) clustering. Obviously, the partitions on the two dimensions have the same number of clusters and it is not possible to specify different numbers of clusters in the two dimensions. Even if interested in the clustering results in a single dimension, the clustering of both rows and columns in a unique partition can be useful to interpret the clustering results: For instance, applying the spectral co-clustering algorithm to a document/term matrix, the resulting clusters contain sets of similar documents and the words characterizing each set of documents, making easier to extract the main concepts/topics associated to each document cluster.

In some application scenarios, such as co-clustering of gene expression data, the requirement of specifying the same number of clusters on both the dimensions of the data matrix (genes and conditions) is considered too strict and impractical. For this reason, Kluger et al. [97] propose a slightly different spectral co-clustering algorithm, tailored for this kind of data. It consists in three consecutive steps: First, the matrix $\mathcal{A}$ is normalized to make the checkerboard pattern more obvious; then, the first few singular vectors are computed, just as in Reference [53]; finally, the eigenvectors are post-processed to find the two partitions.

The idea of using a spectral relaxation of the discrete optimization problem of co-clustering led to the definition of other spectral co-clustering methods [102, 114, 138], although none of these methods have achieved the same popularity of the two reviewed above.

4.2 Nonnegative Matrix Factorization–based Co-clustering

The relation between **Nonnegative Matrix Factorization (NMF)** and one-side clustering has been extensively studied (some reviews of the topic do exist; see for instance Reference [168]). NMF factorizes an input nonnegative matrix into two nonnegative matrices of lower rank. Thus, given a matrix $\mathcal{A} \in \mathbb{R}_+^{n \times m}$, the task is to find two matrices $W \in \mathbb{R}_+^{n \times k}$ and $H \in \mathbb{R}_+^{m \times k}$ such that

$$\mathcal{A} \approx WH^T. \tag{8}$$

The product of the two low rank matrices is not required to exactly equal the input matrix $\mathcal{A}$, but it should approximate it as closely as possible. Thus, a cost function is necessary to quantify the quality of the approximation; the solutions W and H are found by solving a minimization problem of such a function. The most common cost function is the **sum of squared errors (SSE)** $||\mathcal{A} - WH^T||^2$, but other functions such as the I-divergence have been used as well.

NMF provides a general framework for one-side clustering: The columns of H can be considered as k centroids corresponding to k clusters, and each row of matrix $\mathcal{A}$ can be assigned to the closest centroid. However, it has been noticed that the basic NMF provides almost casual clustering [30]; hence, the NMF-based clustering methods usually impose additional orthogonality constraints to W or H to obtain more realistic and interpretable clusters. It can be proved that NMF, with opportune cost objective functions and some orthogonality conditions on the factors W and H, is equivalent to well-known clustering problems [55]: For instance, when the cost function is the SSE, H-orthogonal NMF is equivalent to k-means clustering, orthogonal symmetric NMF is equivalent to Kernel k-means when applied directly to the data matrix, while it is equivalent to spectral clustering when applied to the adjacency matrix of a graph. NMF with I-divergence objective function, instead, is equivalent to Probabilistic Latent Semantic Analysis.

As explained above, NMF is used to cluster the rows of a nonnegative matrix $\mathcal{A}$ and the columns of H are interpreted as the cluster centroids. However, if one is interested in column clustering instead of row clustering, then the result can be obtained by solving the same problem (8), possibly by changing the matrix on which the orthogonality constraints are imposed. At the end, it is sufficient to interpret the columns of matrix W as column clusters' centroids. Given its intrinsic propensity to clustering both rows and columns, and given the excellent results obtained in one-side clustering, the attempt to extend the use of NMF to co-clustering came naturally. Simultaneous clustering of both rows and columns of matrix $\mathcal{A}$ could be done by requiring both H and W to be orthogonal. However, this double constraint is considered too restrictive, resulting in poor low-rank approximation of the data matrix [30]. Instead, it is possible to formulate the co-clustering problem in terms of **Nonnegative Matrix Tri-Factorization (NMTF)**, also known as **Nonnegative Block Value Decomposition (NBVD)**. These two equivalent co-clustering approaches were independently presented by Ding et al. [56] and Long et al. [115] and consist in the factorization of a data matrix $\mathcal{A} \in \mathbb{R}_+^{n \times n}$ in three factors $W \in \mathbb{R}_+^{n \times k}$, $B \in \mathbb{R}_+^{k \times l}$, and $H \in \mathbb{R}_+^{m \times l}$, i.e.,

$$\mathcal{A} \approx WBH^T. \tag{9}$$

While Reference [115] only asks the factors to be nonnegative and considers the rows of W and H as cluster indicator vectors for rows and columns, respectively (it means that the ith row is assigned to the cluster corresponding to the maximum value of W_i, i.e., $\mathcal{R}(i) = \text{argmax}_j W_{ij}$, where $\mathcal{R}$ is the row partition), Reference [56] also imposes orthogonality constraints on W and H: The optimization problem becomes

$$\min_{W \geq 0, B \geq 0, H \geq 0} ||\mathcal{A} - WBH^T||^2, \text{ subject to } W^T W = I, H^T H = I. \tag{10}$$

The columns of W and H can be interpreted as the centroids of the columns and row clusters, and each row (and column) can be assigned to the closest centroid.

In past years, many other variants of NMTF-based co-clustering have been developed: Reference [178] proposes an orthogonal nonnengative matrix tri-factorization algorithm with multiplicative updates; in Reference [162], the factor matrices are constrained to be cluster indicator matrices; other methods impose sparsity constraints on the factor matrices [155] or introduce some form of regularization to reduce the sensitivity of the model to the noise [52]; others focus on the efficiency of the algorithm and propose scalable and/or parallel implementations of NMTF [41, 45]. For other NMTF-based co-clustering methods, the reader is referred to Reference [112].

A further extension of NMTF-based methods that considers geometric aspects of the data is worth mentioning. As noted by Reference [76], the high-dimensional space most real data are represented in could be considered as an embedding of a nonlinear low-dimensional manifold. Hence, both rows and columns of a matrix can be considered as discrete samplings from some manifolds. A way to address this problem is by considering the geometric structure of the manifolds requiring that row and column cluster labels are smooth w.r.t. their respective manifolds. To address this problem, Gu extends orthogonal non-negative matrix tri-factorization by introducing graph regularization for both row and column k-nearest neighbors graphs [76]. Regularization is introduced by adding two regularization terms to Equation (9), computed according to the graph Laplacians of the row and column k-NN graph. The optimization problem then becomes:

$$\min_{W \geq 0, H \geq 0} ||\mathcal{A} - WBH^T||^2 + \lambda \mathrm{tr}(H^T L_H H) + \mu \mathrm{tr}(W^T L_W W), \tag{11}$$

where L_H and L_W are the graph Laplacians of the row k-NN graph and of the column k-NN graph, respectively. Notice that the objective function does not impose any constraint on the positivity of B. Consequently, the algorithm can be applied to general data. Furthermore, the method does not require the orthogonality constraints on W and H.

Other approaches have been defined to deal with manifolds and the geometric structure of the data in general. For instance, the authors of Reference [106] approximate the problem using a convex combination of some candidate manifolds, thus improving the co-clustering performance via manifold ensemble learning. In Reference [9], the authors further elaborate on this idea and propose a multi-manifold matrix tri-factorization algorithm for co-clustering that exploits the geometric structures of the row and column manifolds simultaneously and combines various manifold-based dimensionality reduction methods, such as **multi-dimensional scaling (MDS)**, **isometric feature mapping (ISOMAP)**, and **stochastic neighbor embedding (SNE)**. Finally, in Reference [157], the authors extend Gu's work by introducing the orthogonality constraints on the factor matrices, thus improving the efficiency of the co-clustering algorithm.

4.3 Model-based Co-clustering

Many data analysis methods rely on probabilistic models, i.e., they assume a suitable model for the data generation process and estimate model parameter values from the data. In the context of data clustering, it is assumed that the data arise from a mixture of k underlying probability distributions, where each component of the mixture represents a cluster. Hence, the rows of the data matrix are assumed to be i.i.d. and generated from a probability distribution with density

$$f(x;\theta) = \sum_{c=1}^{k} \pi_c f_c(x;\alpha_c), \tag{12}$$

where f_c is the density of the cth component and it depends on a set of parameters α_c; generally, the densities $f_1, \ldots, f_k$ belong to the same parametric family. π_c is the probability that an observation belongs to the cth component and it is assumed to be known. Thus, the distribution f depends on a set of parameters $\theta = (\pi_1, \ldots, \pi_k, \alpha_1, \ldots, \alpha_k)$, and the goal of a model-based clustering algorithm is to infer the value of these parameters from the data. This is usually done by maximizing the likelihood of θ via the EM algorithm.

As for the spectral and matrix factorization–based clustering, also the model-based clustering approach has been extended to provide a simultaneous partitioning of rows and columns of a matrix. In this case, it is assumed that the data matrix is generated by a probabilistic model in the class of the so-called **Latent Block Models (LBM)**.

In the Latent Block Model [71], each row of a matrix $\mathcal{A}$ is assumed to be a sample from a mixture of k probability distributions and, analogously, each column is a sample from a mixture of l probability distributions. Let us denote with f the distribution of the rows and g the distribution of the columns, both defined as in Equation (12). Each entry a_{ij} of matrix $\mathcal{A}$ can be interpreted as a sample from the joint distribution of rows and columns. Under the hypothesis of conditional independence between rows and columns, the entries of $\mathcal{A}$ are i.i.d. and generated from probability with density

$$\begin{aligned} h(a_{ij};\theta) = h((x_i, y_j);\theta) &= f(x_i;\gamma)g(y_j;\lambda) \\ &= \left(\sum_{r=1}^{k} \pi_r f_r(x_i;\alpha_r)\right)\left(\sum_{c=1}^{l} \pi_c f_c(y_j;\alpha_c)\right) \\ &= \sum_{(r,c)} \pi_r \pi_c f_r(x_i;\alpha_r) f_c(y_j;\alpha_c), \end{aligned} \tag{13}$$

depending on a parameter $\theta = ((\pi_r)_{r=1}^{k}, (\pi_c)_{c=1}^{l}, (\alpha_r)_{r=1}^{k}, (\alpha_c)_{c=1}^{l})$, where x_i and y_j are, respectively, the ith row and the jth column of $\mathcal{A}$. In Reference [71], the authors solve the problem using the classification likelihood approaches aimed at maximizing the following classification log-likelihood associated to the block mixture model:

$$\begin{aligned} L(A, X, Y;\theta) &= \log h(A, X, Y;\theta) \\ &= \sum_r \sum_i u_{ir} \log \pi_r + \sum_c \sum_j v_{jc} \log \pi_c \\ &\quad + \sum_r \sum_i u_{ir} \log f_r(x_i;\alpha_r) + \sum_c \sum_j v_{jc} \log f_c(y_j;\alpha_c), \end{aligned} \tag{14}$$

where u_{ir} and v_{jc} are the entries of the cluster incidence matrices for rows and columns, respectively, i.e., $u_{ir} = 1$ iff row i is assigned to cluster r (0 otherwise), and v_{jc} is defined accordingly.

Several model-based co-clustering algorithms have been proposed, differing in the family of distributions considered for the Latent Block Model; the choice of the distribution, in turn, depends on the type of the data matrix. Typical choices are the Bernoulli distribution for binary data matrices [71, 103], Gaussian distribution for continuous data [119], Poisson, [73] and Multinomial [47, 95] for contingency tables, but also other distributions, such as the BOS distribution [90] and the von Mises-Fisher distribution [147]. The Latent Block model has also been extended to handle mixed type datasets, obtaining the so-called **Multiple Latent Block Model (MLBM)** [149], in which each component of the mixture model has a different probability distribution.

The model-based co-clustering algorithms also differ for the algorithm chosen for the parameter estimation, usually EM-like procedures to find an approximate solution of the maximum likelihood inference problem: The three most used approximation schemes are Variational EM [74], Classification EM [74], Stochastic EM with Gibbs sampler [95], and Stochastic Variational EM [140].

4.4 Information Theoretic Co-clustering and Similar Approaches

In this section, we present a class of co-clustering algorithms based on the iterative optimization of some quality measure for the co-clustering solution. Informally, the idea behind all these methods is that a co-clustering $(\mathcal{R}, \mathcal{C})$ over a data matrix $\mathcal{A}$ is good if the column partition $\mathcal{C}$ is useful to explain or describe the row partition $\mathcal{R}$ and vice versa. The functions used to quantify the strength of the relation between the row and column partitions are usually borrowed from the field of Information Theory.

Suppose there are two categorical variables, X with k possible values and Y with l possible values. Let p_X and p_Y be the probability distributions of X and Y, respectively, and p_{XY} their joint probability. There are several ways to measure the association between X and Y, i.e., the dependence between the two variables. Early approaches, such as the one of Hartigan [78] and Govaert [70], adopts the least-squared criterion, similar to the k-means objective in one-mode clustering. Bock, instead, adopts a variance-based criterion [23], similar to the one used by Cheng and Church for their biclustering algorithm [42]. More recent approaches use information-theoretic functions, such as the mutual information between X and Y [48] that quantifies the amount of information obtained about one random variable by observing the other, and it is defined as

$$I(X,Y) = \sum_{i=1}^{k}\sum_{j=1}^{l} p_{XY}(i,j)\log\left(\frac{p_{XY}(i,j)}{p_X(i)p_Y(j)}\right). \tag{15}$$

Another well-known measure is the Φ^2 Pearson's coefficient of mean squared contingency [132]:

$$\Phi^2(X,Y) = \sum_{i=1}^{k}\sum_{j=1}^{l} \frac{(p_{XY}(i,j) - p_X(i)p_Y(j))^2}{p_X(i)p_Y(j)}. \tag{16}$$

Both these association measures are symmetric, i.e., $I(X,Y) = I(Y,X)$ and $\Phi^2(X,Y) = \Phi^2(Y,X)$. Instead, Goodman and Kruskal's τ [69] is an asymmetric measure that quantifies the association of X to Y measuring the proportional reduction of the error in predicting X when Y is known. It is defined as

$$\tau_{X|Y}(X,Y) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{l}\frac{p_{XY}(i,j)^2}{p_Y(j)} - \sum_{i=1}^{k} p_X(i)^2}{1 - \sum_{i=1}^{k} p_X(i)^2}. \tag{17}$$

Analogously, the association of Y to X is the proportional reduction of the error in predicting Y when X is known and is defined as

$$\tau_{Y|X}(X,Y) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{l}\frac{p_{XY}(i,j)^2}{p_X(i)} - \sum_{j=1}^{l} p_Y(j)^2}{1 - \sum_{j=1}^{l} p_Y(j)^2}. \tag{18}$$

All these measures (and possibly others) can be used to evaluate the quality of a co-clustering solution over a contingency table $\mathcal{A} \in \mathbb{R}_+^{n\times m}$. Since the input matrix $\mathcal{A}$ is a co-occurrence table between a set of n categorical values (one per row) and m other categorical values (one per column), we can interpret our data as a sample S of $\sum_{ij} a_{ij}$ objects from a population; the objects are described by two different categorical attributes, the first with n classes and the second with m classes. Hence, if we denote by X the random variable reporting the value of the first attribute for a randomly extracted object and Y the random variable reporting the value of the second attribute, table $\mathcal{A}$ is the contingency table of the two categorical random variables X and Y over the sample S. Giving a co-clustering of matrix $\mathcal{A}$ means finding a partition of the values of X and another partition of the values of Y. Let $\mathcal{R}$ and $\mathcal{C}$ be these partitions. They correspond to a new pair of random variables, R and C, where R returns the cluster in partition $\mathcal{R}$ of a randomly picked object and C returns the cluster of an object according to partition $\mathcal{C}$. The contingency table of R and C on the same sample S is $\mathcal{T} = \mathcal{T}(\mathcal{A},\mathcal{R},\mathcal{C})$, whose generic entry is

$$t_{rc} = \sum_{i|\mathcal{R}(i)=r}\;\sum_{j|\mathcal{C}(j)=c} a_{ij} \tag{19}$$

and the probability associated to R, C and (R, C) are, respectively, $p_R(r) = \frac{\sum_{c=1}^{|C|} t_{rc}}{T}$, $p_C(c) = \frac{\sum_{r=1}^{|\mathcal{R}|} t_{rc}}{T}$ and $p_{RC}(r, c) = \frac{t_{rc}}{T}$, where $T = \sum_{r=1}^{k} \sum_{c=1}^{l} t_{rc}$. Thus, the quality of the co-clustering $(\mathcal{R}, C)$ can be quantified by measuring the association between the random variables R and C, using one of the association measures introduced above. Different association measures lead to different co-clustering algorithms: **Information Theoretic co-clustering (ITCC)** [54] maximizes the mutual information $I(R, C)$; an algorithm maximizing $\Phi^2(R, C)$ is given in Reference [74]; a family of algorithms that simultaneously maximize $\tau_{R|C}$ and $\tau_{C|R}$ has been presented in References [88, 135, 141]. In these papers, the co-clustering problem is formulated as an optimization problem of the respective objective function, solved by an alternating optimization of partition $\mathcal{R}$, keeping C fixed, and then of partition C, keeping $\mathcal{R}$ fixed.

For convenience, in this class of methods, we insert also other algorithms that use the same alternated optimization schema but consider objective functions that are not probabilistic measures of association. For instance, Reference [6] uses an objective function usually employed in graph clustering, the Newman's Modularity [120]. The modularity can be used to measure the quality of a diagonal co-clustering $(\mathcal{R}, C)$ over a contingency table $\mathcal{A}$. A co-clustering is diagonal if the row and column partitions have the same number of clusters, i.e., $|\mathcal{R}| = |C| = k$. In the co-clustering context, modularity is defined as

$$Q(\mathcal{A}, \mathcal{R}, C) = \frac{1}{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}} \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{h=1}^{k} \left(a_{ij} - \frac{\sum_{j=1}^{m} a_{ij} \sum_{i=1}^{n} a_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}} \delta_{ih}^{R} \delta_{jh}^{C} \right), \tag{20}$$

where $\delta_{ih}^{R} = 1$ if $\mathcal{R}(i) = h$ and 0 otherwise, and analogously $\delta_{jh}^{C} = 1$ if $C(j) = h$ and 0 otherwise.

Another algorithm that can be included in this class of co-clustering methods is the one proposed in Reference [44], which is a sort of generalization of the Lloyd's algorithm for k-means: It adopts, as objective function, the sum of the squared residues, where the residue of element a_{ij} is the distance between a_{ij} and the co-cluster to which a_{ij} is assigned. However, although this method follows an optimization schema similar to the other algorithms mentioned in this section, it is designed to work with continuous data.

We will now examine in more detail how the optimization scheme of these algorithms works. Excluding the approach based on the optimization of Goodman and Kruskal's τ, which will be discussed later, all other algorithms have a similar "k-means like" structure: They start with an initial co-clustering $(\mathcal{R}^0, C^0)$ with k and l clusters, respectively (when required, $k = l$), and compute a cluster prototype for each row cluster in $\mathcal{R}$. Then, using an appropriate function to measure the similarity between rows and cluster prototypes, they assign each row to the closest prototype. Next, the same procedure is applied to the columns to update the column clustering. The way in which the cluster prototypes are computed and the similarity measure are different for each algorithm: They are opportunely designed to guarantee that the objective function increases at each iteration.

Instead, the co-clustering method based on the optimization of τ, which we will call τCC from now on, rather than optimizing a unique objective function, maximizes two objective functions at the same time, $\tau_{R|C}$ and $\tau_{C|R}$. Thus, co-clustering is formulated as a multi-objective optimization problem, and a stochastic local search approach is used to solve each maximization problem: At each iteration, only one row and one column are moved from their original cluster to a new, more suitable, cluster: This is different from the other methods that, at each iteration, re-assign all the rows and all the columns using a prototype-based optimization. Furthermore, the objective function τ has an upper bound that does not depend on the number of clusters. This fact enables algorithm τCC to compare co-clustering configurations with a different number of clusters and choose the best one: As a consequence, τCC is able to identify the final number of clusters on its own. Recently, a de-normalized version of τ has been used in Fast-τCC [18], a faster and considerably more scalable version of τCC adopting a prototype-based optimization strategy similar to the one of ITCC.

4.5 Other Co-clustering Methods

There are other co-clustering approaches that do not fall into any of the above classes. Recently, a deep learning model for co-clustering called DeepCC was proposed in Reference [171]. It uses two separated autoencoders to reduce the dimensionality of both the rows and the columns and then two separated inference networks to infer the parameter of two **Gaussian Mixture Models (GMM)**, one for the rows and one for the columns. The autoencoders and the inference networks are trained together: DeepCC jointly minimizes the reconstruction error of the deep autoencoder and maximizes the variational lower bound of the log-likelihood in GMM. Furthermore, an instance-feature cross loss term based on Mutual Information is added to the objective function to make the training of instances and features intertwined. The structure of the overall deep model (i.e., the number of levels, the number of neurons for the two autoencoders and the two inference networks, and the activation function) are parameters required by the algorithm, together with three weights for the different terms composing the objective function. These parameters also include the number of clusters on rows and columns, corresponding to the number of neurons in the last level of the inference network.

A different approach is **co-clustering through Optimal Transport (CCOT)**, presented in Reference [104]. The paper poses the co-clustering problem as the task of transporting the empirical measure defined on the data instances to the empirical measure defined on the data features. Given two sets of objects X and Y and their empirical discrete probability measures $\mu_X = \frac{1}{|X|}\sum_{i=1}^{|X|}\delta_{x_i}$ and $\mu_Y = \frac{1}{|Y|}\sum_{j=1}^{|Y|}\delta_{y_j}$, the optimal transportation problem consists in finding a coupling γ, defined as a joint probability measure over $X \times Y$ with marginals μ_X and μ_Y, minimizing the cost of transport. In the context of co-clustering, X and Y are, respectively, the rows and the columns of a squared matrix $\mathcal{A} \in \mathbb{R}_+^{n\times n}$, and the cost of transport is defined as

$$T_\lambda(\gamma) = Tr(M\gamma^T) - \frac{1}{\lambda}E(\gamma), \tag{21}$$

where M is a dissimilarity matrix computed as $m_{ij} = ||x_i - y_j||^2$, λ is a positive real number, and $E(\gamma) = -\sum_{i,j}\gamma_{ij}\log(\gamma_{ij})$ is the entropy of matrix γ. It can be proved that there exists a unique solution γ_λ^* minimizing Equation (21) and has the form

$$\gamma_\lambda^* = diag(\alpha)\xi_\lambda diag(\beta),$$

where $\xi_\lambda = e^{-\lambda M}$ is the Gibbs kernel, and α and β are unique up to a multiplicative factor. The solutions α and β can be found using using the Sinkhorn-Knopp algorithm.

Notice that, although formulated in terms of an optimization problem, CCOT is different from the methods presented in the previous section: In fact, its objective function (see Equation (21)) does not depend on the co-clustering solution but only on the input data matrix (through the dissimilarity matrix M). The result of the optimization is a coupling matrix γ^*, which is later factorized into three terms where one of them, ξ_λ, reflects the posterior distribution of the data given the co-clusters, while two other terms, α and β, represent the approximated distributions of the data instances and features, respectively. These approximated distributions are used to obtain the final partitions. An interesting property of this co-clustering approach is that the method used to extract the clustering assignment from vectors α and β automatically detects the final number of clusters, using a procedure (originally introduced for multiscale denoising of piecewise smooth signals) to detect steps in the approximated distributions α and β. The rows (respectively, columns) corresponding to the values of α (respectively, β) lying between two consecutive steps belong to the same cluster. Thus, the algorithm does not need the specification of the number of clusters k and l as input parameters. However, the objective function (Equation (21)) depends on a parameter λ; moreover, since the proposed optimization method only applies to matrices with the same number of rows and columns, to work with generic rectangular matrices, the algorithm involves a first step in which n_s squared matrices are subsampled from the data and then the co-clustering procedure is applied to each one of the sampled matrices. The final clustering results are obtained through the majority vote over all samples. The number of samples n_S must be large enough to guarantee that each row and column of the original data matrix is picked in at least one sample and is a further parameter of the co-clustering algorithm. To avoid this, the authors also proposed a kernelized version of their algorithm, called CCOT-GW, based on the notion of Gromov-Wasserstein distance.

Two common problems of the optimal transport-based approaches are that co-clustering is not the main objective of the optimization process and that they have high computational costs, in terms of both memory and time complexity. A recent work [61] addresses these issues by proposing an algorithm, called **BCOT (Biclustering using Optimal Transport)**, that integrates the co-clustering objective from the beginning, thus providing a general formulation to several co-clustering methods. The problem is then formulated as a bilinear program, solved using an efficient block coordinate descent algorithm where the row assignments are computed by fixing the column assignments, and then vice versa iteratively, until convergence is achieved. The optimal transport for each row and column assignment step is formulated as the **Earth Mover's Distance (EMD)** problem, optimized using a minimum-cost flow algorithm.

To conclude this section, we mention a bunch of methods to perform ensemble co-clustering [2, 84, 180]. The idea exploited by these methods is that different co-clustering algorithms grasp different aspects from the data and, thus, combining different co-clustering algorithms should lead to better results in terms of consistency and quality. The key and most difficult issue in ensemble (co-)clustering is how to combine the clusters that are generated by the individual clustering models, as this cannot be done through simple voting or averaging as in classification. Reference [84] proposes a **spectral co-clustering ensemble algorithm (SCCE)** formulated as a double graph partition problem: The vertices of the graph are the rows and columns of the input matrix, and the weight of the edge between two nodes (rows or columns) depends on how many co-clustering solutions in the ensemble put the nodes in the same cluster. The final co-clustering solution is computed via a spectral clustering of the graph. Also, the algorithm in Reference [180] uses a spectral co-clustering of a graph representation of the multiple co-clustering results, but, differently from the previous method that only considers row-to-row and column-to-column edges, the resulting graph also exploits the row-to-column relations captured by the different co-clustering solutions.

Instead, in Reference [2], the problem of combining multiple co-clustering solutions is modeled as an optimization problem, solvable through the factorization of an affinity matrix created by combining the contingency matrices of all the co-clustering solutions of the ensemble. Interestingly, the method can be used to combine co-clustering solutions with different number of co-clusters, but the number of desired co-clusters in the final solution is passed to the algorithm as a parameter. Furthermore, this method only returns diagonal co-clustering solutions (i.e., solutions with the same number of clusters on rows and columns).

5 Theoretical Comparison of the Main Co-clustering Methods

In this section, we compare the main categories of co-clustering algorithms according to different theoretical and practical aspects. More in detail, we provide a characterization of the algorithms according to their theoretical aspects, including time complexity. In the Supplemental Material (Appendix B), we also include an experimental comparison of the main categories of co-clustering methods to help the practitioners assess the applicability of each method to their own problems.

Although classified in several groups for the sake of clarity, the various co-clustering methods presented in this section are often related to each other and present more similarities than the strict classification given above would suggest. For instance, the relation between spectral co-clustering methods and nonnegative matrix factorization is quite evident, since both are based on some matrix decomposition: An analysis of the relationship between the two methods is given in Reference [55]. Instead, in Reference [75] the authors propose a unified theoretical framework into which Information Theoretic co-clustering algorithms as ITCC and Φ^2-CC and model-based algorithms such as the Poisson Latent Block Model can be incorporated.

Here, we report a comparison of the mentioned co-clustering models, focused on the parameters and assumptions on the data generation process they need to be applied. As seen, the vast majority of co-clustering methods require the number of clusters on rows and columns to be set in advance. Setting a different number of clusters can lead to very distant co-clustering solutions. However, in realistic application scenarios, the actual number of clusters is not known. Sometimes, the choice of a congruent number of clusters is done using the computationally expensive exhaustive search over a large number of possible pairs of row and column clusters: For model-based co-clustering, Reference [95] gives a model selection criterion based on the maximization of the Integrated Completed Likehood; in Reference [6], the authors argue that the algorithm ModCC they propose, based on the optimization of the Modularity (Equation (20)), can be used to identify the correct number of co-clusters embedded in the data matrix. In fact, Modularity is a function with values in $[0, 1]$ and its upper bound does not depend on the number of clusters. Thus, the number of co-clusters can be selected by applying algorithm ModCC several times changing the value of the parameter k (the number of co-clusters) and, finally, choosing the number k corresponding to the co-clustering solution with highest Modularity. Notice that, despite its objective function does not depend on the number of clusters, ModCC is not a parameter-free algorithm, i.e., it requires the specification of the correct number of clusters. Furthermore, ModCC can only return diagonal co-clustering solutions. The Modularity criterion for the selection of k is used also in the co-clustering ensemble method [2] to infer the number of co-clusters of the consensus solution. A bunch of alternative approaches for the evaluation of latent block model-based and non-negative matrix tri-factorization–based co-clustering have been developed within the specific domain of word-document co-occurrence matrices [2, 8, 144].

The repeated application of the same co-clustering algorithm varying the input parameters k and l with the goal of selecting the parameter configuration that maximizes the objective function is not always applicable. This is due to the fact that, usually, the magnitude of the co-clustering objective functions strongly depends on the number of clusters. Consequently, two solutions with two different numbers of clusters can not be compared directly. This is, for instance, the case of the mutual information used in ITCC: It is always maximum for the discrete partition, i.e., when each cluster contains exactly one data instance. For this kind of algorithms, the only available way to identify a realistic number of clusters is to use some exploratory data analysis method, such as the Elbow method or the Silhouette method. These criteria are developed to select the number of clusters in one-side clustering, but can be used on rows and columns separately to find an appropriate value for k and l.

The only two methods that do not require the specification of the number of clusters are τCC (see Section 4.4) and CCOT (see Section 4.5). The former maximizes an objective function τ whose optima do not depend on the number of clusters; instead, the latter compute an approximate distribution for the rows and columns of the input matrix and then use a procedure to detect jumps

Table 1. Summary of the Characteristics of the Main Co-clustering Methods for Contingency Tables, with a Focus on the Parameters Required and the Time Complexity

Algorithm	Diagonal ($k = l$)	Requires k and l	Further parameters and other assumptions	Obj. Function enabling comparison of different (k, l)	Time complexity
SpectralCC [53]	Yes	Yes	# of eigenvectors (optional)	-	$O(mnt \log(r))$ (dense) $O(zt \log(r))$ (sparse)
SpectralBiC [97]	No	Yes	Normalization # of eigenvectors	-	$O(mnt \log(r))$ (dense), $O(zt \log(r))$ (sparse)
NMTF [56, 115, 162]	No	Yes	Factorization rank (optional)	-	$O(tm^2n)$ [56], $O(mnt(k+l))$ [115], $O(nmkt)$ [162]
LBM [72, 73, 95, 103]	No [72, 73, 95] Yes [103]	Yes	Distribution family	Yes	$O(mnklt)$ (dense), $O(zklt)$ (sparse)
ITCC [54]	No	Yes	None	No	$O(mnt(k+l))$ (dense), $O(zt(k+l))$ (sparse)
Φ^2-CC [73]	No	Yes	None	No	$O(mnt(k+l))$
ModCC [6]	Yes	Yes	None	Yes	$O(mntk)$ (dense), $O(ztk)$ (sparse)
τCC [88, 141]	No	No	None	Yes	$O(tmn)$
Fast-τCC [18]	No	No	None	Yes	$O(tkl(m+n))$
DeepCC [171]	No	Yes	Network structure weights of the loss	-	$O(tn(m+P))$
CCOT [104]	No	No	λ, n_s (cf. Section 4.5)	-	$O(n_s(n+m))$ (CCOT) $O(n^2m + m^2n)$ (CCOT-GW)
BCOT [61]	No	Yes	Row/column weights, exemplar distributions	Yes	$O(kz + t(n+k)nk \log(n+k))$

k and l indicate, respectively, the number of clusters on the rows and on the columns. n and m are the dimension of $\mathcal{A}$, while z is the number its non-zero entries and r is the number of eigenvalues. t is the total number of iterations and P is the number of trainable parameters of the deep neural network.

in the approximated distributions. However, as pointed out before, CCOT depends on two further parameters.

As CCOT, many of the algorithms considered in this review are based on some assumptions, which consist in further parameters for the model: The most demanding algorithm in this sense is DeepCC, which, as seen in Section 4.5, requires the setting of several hyper-parameters; in the spectral methods, a further parameter is the number of retained eigenvectors (many algorithms, however, set this number equal to $\log_2 n$ or equal to k); in the model-based methods, the choice of the distribution used to model the data is a strong assumption and different distributions lead to different co-clustering solutions; in the matrix factorization–based algorithms, sometimes the shape of the latent factors is left as a model parameter (when set different from the number of clusters, a final run of $k(l)$-means is used to recover the clustering assignments). Table 1 summarizes the main characteristics of the main co-clustering described in this section. We also report the time complexity of each category of algorithm, as reported by the authors themselves in the respective papers or inferred by us. In general, the vast majority of algorithms (with the exception of CCOT [104] and orthogonal NMTF [56]) scales linearly w.r.t. the number of rows (n) and columns (m) of the input matrix or the number of non-zeros (z). Those algorithms whose complexity also depends on the number of clusters, in general, scales linearly w.r.t. the sum of k and l or the largest of the two values. The time complexity of the spectral methods, instead, is in the logarithm of r, where r is the number of eigenvalues as computed in the SVD step [53, 97]. A more in-depth discussion is required for the number of iterations, as almost all algorithms consist of iterative steps. The main difference is the semantic of an individual iteration. While many algorithms alternate the optimization of the row partition and the optimization of the column one, for other algorithms, the number of iterations has different meanings. For instance, in *LBM* [72, 73, 95] and Fast-τCC, the overall number of iterations depends on the number of inner optimization steps, multiplied by the number of outer steps, while in τCC [88, 141] it is the overall number of moves of a single row or column, which is typically some orders of magnitude higher than the typical number of iterations required in other algorithms.

Finally, it is worth spending a few words on the time complexity of DeepCC [171], the only approach based on a deep learning architecture. In this case, the complexity depends on stochastic gradient descent, whose single iteration on an individual data sample is linear in the number of overall parameters of the network (noted with P, in Table 1, according to Reference [25]. However, P could be as large as several hundred million parameters in rather small matrices.

6 Tensor Co-clustering

The majority of the data produced by human activities and modern cyber-physical systems involves complex relations among their features. Such relations can often be represented by means of tensors, widely used mathematical objects that well represent complex information such as gene expression data [185], social networks [82], heterogenous information networks [59, 179], time-evolving data [11], behavioral patterns [79], and multi-lingual text corpora [129].

Tensors can be viewed as generalizations of matrices and, as such, can be analyzed by using higher-order extensions of existing machine learning methods, many of which are tensor decomposition models and algorithms [98]. As an example, both singular value decomposition [183] and non-negative matrix factorization [150] have been extended to work with high-order tensor data. Furthermore, knowledge discovery and exploratory data mining techniques, including closed itemset mining [33, 34] and association rule discovery [121], have been successfully applied to N-way data as well.

Also, clustering and co-clustering have been naturally extended to handle tensors. In this section, we briefly review the main tensor co-clustering method, but, before that, we give a formal definition of what a tensor is and which kind of relations among the data it is able to capture.

6.1 Tensors vs. Multi-view Datasets

Usually, data come in the form of a set of objects described by a set of attributes (also called features). These features can be heterogeneous for type and underlying properties. When the features are all numeric, the dataset consists in a matrix $\mathcal{A} \in \mathbb{R}^{n \times m}$, where n is the number of objects and m the number of features. However, in this survey, we only consider datasets with homogeneous attributes: Only in this case, in fact, the problem of clustering the columns of the matrix makes sense. Another way to define this kind of matrices is as mathematical objects used to describe binary relation between two sets of discrete objects, $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$. From this perspective, a matrix $\mathcal{M}$ is the co-domain of a function $m : A \times B \longrightarrow \mathbb{R}$, i.e., $m_{ij} = m((a_i, b_j))$. For instance, we obtain a document/terms matrix from the function $m : D \times T \longrightarrow \mathbb{R}$, where D is a set of documents and T a dictionary of terms, that associates to each pair (d, t) the number of occurrences of the term t in document d. As another example, a gene expression matrix is the co-domain of the function $e : G \times C \longrightarrow \mathbb{R}$, where G is a set of genes and C a set of conditions, sending the pair (g, c) to the expression level of the gene g under condition c.

When there exist multiple relations between the two sets of objects A and B, a simple matrix is no longer sufficient to represent the data. The problem becomes even more complicated when the sets of objects are more than two, and multiple relations among all or part of these sets are present. Such a complex scenario requires more sophisticated data representation: two possible extensions of data matrices to handle multiple relations among sets of objects are tensors and multi-view datasets.

Tensors and multi-view datasets are different objects representing different relations among several sets of objects. Given three sets A, B, and C with, respectively, n, m, and l objects, a 3-way tensor with shape $n \times m \times l$ can be used to describe a measurable relation among all the possible triples of objects (a_i, b_j, c_k). More formally, a tensor $\mathcal{T} \in \mathbb{R}^{n\times m\times l}$ is the co-domain of a function

$$t : A \times B \times C \longrightarrow \mathbb{R},$$

i.e., $\mathcal{T} = t(A \times B \times C)$, and its generic entry is $t_{ijk} = t((a_i, b_j, c_k))$. Importantly, $\mathcal{T}$ represents a ternary relation over $A \times B \times C$. Instead, a multi-view dataset can be used to express multiple binary measurable relations. For instance, two functions

$$g : A \times B \longrightarrow \mathbb{R},$$

$$h : A \times C \longrightarrow \mathbb{R}$$

define a multi-view dataset $D = (\mathcal{G}, \mathcal{H})$, where $\mathcal{G} \in \mathbb{R}^{n\times m_g}$ and $\mathcal{H} \in \mathbb{R}^{n\times m_h}$. Their generic elements are $g_{ij} = g((a_i, b_j))$ and $h_{ik} = h((a_i, c_k))$. Notice that a tensor can be re-arranged to become a multi-view dataset, while the contrary is not necessarily true. In fact, given the ternary function t defining tensor $\mathcal{T}$, it is always possible to obtain two binary functions

$$g : A \times B \longrightarrow \mathbb{R}; \qquad (a_i, b_j) \longmapsto \sum_{k=1}^{l} t((a_i, b_j, c_k)),$$

$$h : A \times C \longrightarrow \mathbb{R}; \qquad (a_i, c_k) \longmapsto \sum_{j=1}^{m} t((a_i, b_j, c_k))$$

and thus a multi-view dataset $(g(A \times B), h(A \times C))$. Instead, given two functions $g : A \times B \longrightarrow \mathbb{R}$ and $h : A \times C \longrightarrow \mathbb{R}$, they do not automatically extend to a function over $A \times B \times C$.

Summarizing, an N-way (or N-mode) tensor is used to express an N-ary relation among N sets of objects. Instead, a multi-view dataset is a collection of $N - 1$ matrices and is used to express N different binary relations between one set of items and $N - 1$ other discrete sets. We will use the following notation: Given an N-way tensor $\mathcal{T} \in \mathbb{R}^{n_1\times\cdots\times n_N}$, its generic entry is $t_{i_1\ldots i_N}$, with $i_r = 1, \ldots, n_r$ for each $r = 1, \ldots, N$. Instead, given a multi-view dataset $D = (\mathcal{D}^1, \ldots, \mathcal{D}^{N-1})$, where $\mathcal{D}^r \in \mathbb{R}^{n\times m_r}$ for each $r = 1, \ldots, N - 1$, a generic entry is d^r_{ij}, where r indicates the view, $i = 1, \ldots, n$ indicates the row, and $j = 1, \ldots, m_r$ indicates the column in the rth view. Figure 2 in the Supplemental Material graphically shows the difference between multiple views and tensors.

Going back to the main topic of this article, co-clustering can be extended to work with both these data objects, tensors, and multi-view datasets:

- **Tensor Co-clustering** is the simultaneous partitioning of all the N modes of a tensor. Thus, a co-clustering of an N-way tensor $\mathcal{T}$ is a tuple of N partitions $(\mathcal{P}_1, \ldots, \mathcal{P}_N)$ and, as in the 2-way case, each partition must be exhaustive and exclusive, i.e., each object on each mode $i = 1, \ldots, N$ must belong to one and only one cluster of the ith partition $\mathcal{P}_i$. Notice that also the notion of biclustering introduced at the beginning of this chapter can be extended to tensors: The problem of N-way clustering a tensor $\mathcal{T}$ with shape $n_1 \times \cdots \times n_N$ is the problem of finding a subtensor $\mathcal{A}$ of $\mathcal{T}$ with shape $k_1 \times \cdots k_N$, where $k_i < n_i$ for all i, such that the entries of $\mathcal{A}$ satisfy some criteria of homogeneity or coherence. $\mathcal{A}$ is called an "N-way" cluster and the problem can be generalized to identifying many N-way clusters inside $\mathcal{T}$. Differently from the tensor co-clustering problem, here, the clusters can be overlapping and non-exhaustive. However, the N-way clustering task is out of the scope of this work; for more details and a detailed survey about this topic, see Reference [80].

- **Multi-view Co-clustering** consists in the simultaneous partitioning of the rows and of all the $N-1$ sets of attributes of a multi-view dataset. Thus, also a multi-view co-clustering solution is a set of N partitions. The difference with tensor co-clustering lies in the relationships between the N clustering tasks: While in tensor co-clustering every clustering task benefits from the others, in the sense that a high-quality partition on $N-1$ modes helps in finding a better partition also in the Nth mode, in multi-view co-clustering, having good clustering on the different sets of attributes helps in finding a better partition on the row, but the partition on a particular set of attributes does not play any role in the choice of the partition on the other sets of attributes, or, more exactly, their relationship is only mediated by the row partition.

In the next section, we will go through the main tensor co-clustering algorithms in the literature. Instead, even if many multi-view co-clustering algorithms do exist, they are, usually, extensions of the co-clustering methods for a single matrix: For instance, Reference [172] extends ITCC, Reference [88] is a multi-view co-clustering based on the optimization of Goodman and Kruskal's τ, Reference [85] uses multi-view spectral co-clustering, Reference [122] performs multi-view co-clustering via nonnegative matrix factorization, and Reference [28] adopts an approach based on sparse probabilistic latent block model. We refer the reader to Reference [36], a recent survey on multi-view clustering that reviews some multi-view co-clustering as well.

Actually, multi-view co-clustering is a more prosperous research field than tensor co-clustering. This can probably be explained by the fact that tensors can be rearranged to form multi-view datasets, and then multi-view co-clustering can be used to cluster tensor data. In contrast, many datasets that can be represented with multiple views cannot be expressed in the form of tensors (because they lack a relation between each pair of sets of attributes); hence, multi-view co-clustering is a more generic task that can be applied to a broader range of datasets. However, when dealing with tensors, tensor co-clustering is more suitable than multi-view co-clustering, because it captures more sophisticated relations among all the dimensions describing the data.

6.2 Main Approaches to Tensor Co-clustering

The main tensor co-clustering methods are often based on tensor decomposition models, such as **CANDECOMP/PARAFAC (CP)** [77] or Non-negative Tucker decomposition [158]. These approaches can be seen as an extension of factorization-based and spectral co-clustering methods: In fact, both CP and Tucker decomposition are higher-order generalizations of Singular Value Decomposition. Additionally, the usage of nonnegativity constraints in the tensor decompositions resembles the nonnegative matrix factorization methods.

In more detail, following the notation of Reference [98], the CP decomposition factorizes a tensor into a sum of component rank-one tensors. Given a third-order tensor $\mathcal{T} \in \mathbb{R}^{n\times m\times l}$, it can be rewritten as

$$\mathcal{T} \approx \sum_{r=1}^{R} u_r \circ v_r \circ z_r, \tag{22}$$

where $\circ$ is the matrix outer product, R is a positive integer (the rank of the decomposition), and $u_r \in \mathbb{R}^n$, $v_r \in \mathbb{R}^m$, and $z_r \in \mathbb{R}^l$ for $r = 1 \ldots R$. According to this notation, matrices $U = [u_1\, u_2\, \cdots\, u_R]$, $V = [v_1\, v_2\, \cdots\, v_R]$, and $Z = [z_1\, z_2\, \cdots\, z_R]$ are the rank-R factor matrices.

For the same generic three-mode tensor $\mathcal{T}$, instead, the Tucker decomposition factorizes it into a core tensor $\mathcal{W} \in \mathbb{R}^{P\times Q\times R}$ multiplied by a factor matrix along each mode, i.e.,

$$\mathcal{T} \approx \mathcal{W} \times_1 U \times_2 V \times_3 Z, \tag{23}$$

where $\times_n$ is the n-mode matrix product, and $U \in \mathbb{R}^{n\times P}$, $V \in \mathbb{R}^{m\times Q}$, and $U \in \mathbb{R}^{l\times R}$ are the (usually orthogonal) factor matrices. Additionally, when the input tensor $\mathcal{T}$ is non-negative, non-negativity constraints can be imposed on all the factor matrices and the core tensor as well.

For both CP and Tucker models, the basic tensor co-clustering approaches work similarly: Once the factor matrices are computed using some optimization algorithms, such as the Alternating Least Square or the (Stochastic) Gradient Descent methods, the objects in each mode are assigned by processing the corresponding factor matrix with k-means. Alternatively, each object can be assigned to the cluster corresponding to the index of the largest factor value.

In Reference [186], the authors use tensor-based latent factor analysis to address co-clustering in the context of web usage mining; their algorithm is executed via the CANDECOMP/PARAFAC decomposition. Reference [130] formulates co-clustering as a constrained multi-linear decomposition with sparse latent factors. It consists in a basic multi-way co-clustering algorithm exploiting multi-linearity using Lasso-type coordinate updates. Additionally, a line search optimization approach based on iterative majorization and polynomial fitting is proposed. The authors of Reference [184] present an extension of the Nonnegative Matrix Tri-factorization model [56] to a tensor decomposition model performing adaptive dimensionality reduction by integrating the subspace identification and the (hard or soft) clustering process into a single process. Their algorithm computes two basis matrices representing the common characteristics of the samples and one 3-D tensor denoting the peculiarities of the samples. The model can be used to perform dimensionality reduction as well. Instead, Reference [169] introduces a spectral co-clustering method based on a new random walk model for nonnegative square tensors; finally, in Reference [43], the tensors are represented as weighted hypergraphs, which are cut by an algorithm based on random sampling.

Another class of recent approaches [26, 27] relies on an extension of the Latent Block Model. In these works, co-clustering for sparse tensor data is viewed as a multi-way clustering model where each slice of the third mode of the tensor represents a relation between two sets. The same authors also developed a Python library for tensor clustering and co-clustering, called **TensorClus** [29], including their model-based tensor co-clustering algorithms. Finally, Reference [164] presents a co-clustering approach for tensors that uses a least-square estimation procedure for identifying an N-way block structure that applies to binary, continuous, and hybrid data instances.

In contrast, there are very few works that model tensor co-clustering as an optimization problem of an Information Theoretic measure, extending some of the algorithms presented in Section 4.4: Reference [13] performs clustering using a relation graph model that describes all the known relations between the modes of a tensor. The tensor clustering formulation captures the maximal information in the relation graph by exploiting a family of loss function known as Bregman divergences. The authors also present several structurally different multi-way clustering schemes involving a scalable algorithm based on alternate minimization. A more recent work [65] follows the same research line based on the optimization of a Bregman divergence.

The only other work based on Information Theoretic measures are References [18, 19]: The methods proposed there are extensions of τCC and Fast-τCC, reviewed in Section 4.4, and, as such, they are also parameter-less.

7 Recent Trends and Open Problems

In this section, we discuss the most recent trends in co-clustering research, highlighting the main scientific results of the past three years. Then, we analyze the possible reasons for a perceived decreased interest in designing new co-clustering methods. Finally, we point out some open or scarcely addressed problems and provide some promising research directions that are worth investigating in the next few years.

7.1 Recent Contributions to the Co-clustering Literature

As we noted in the previous sections, in the past five years the only novel approach is the one based on deep neural networks [171]. Although no further groundbreaking contributions have emerged in the past three years, the interests of scientists have significantly shifted towards multi-view approaches, with several proposals aimed at solving this type of problem. This is consistent with an overall trend in machine learning and data science, where complex, multi-relational data have been the object of many studies [66, 175, 182]. Notable examples are References [57, 87]. The first work is an extension of the deep co-clustering framework to work with sparse multi-view data [57]. The authors propose a differentiable network with alternating iterative optimization that they call "differentiable bi-sparse multi-view co-clustering." The second paper combines a k-means strategy with local search to iteratively optimize the co-clustering objective function by updating cluster labels, features, and view weights [87].

Other recent approaches have focused on improving or speeding up matrix factorization algorithms for co-clustering. For instance, in Reference [41], the authors use the Lagrange multipliers to derive the original optimization problem of NMTF and propose a distributed implementation of the algorithm that optimizes the Lagrange dual objective function. Instead, the authors of Reference [167] address the graph co-clustering problem by integrating network embedding and nonnegative matrix factorization, following the idea that graph representation learning implicitly implies matrix factorization. They prove the equivalence between NMF and graph embedding for co-clustering and propose an algorithm that regularizes network embedding into the NMF objective function. In Reference [52], the problem of noise sensitivity of the square loss method for nonnegative matrix factorization is addressed. To solve the problem, the authors combine multiple regularization tricks (e.g., graph regularization, Frobenius norm, and norm) to simultaneously optimize the objective function. The proposed method is also able to perform feature selection well and enhance the sparseness of the model, thus obtaining less noisy data matrices to approximate the objective matrix. The authors of Reference [157], instead, propose an algorithm to solve the orthogonal dual graph regularized nonnegative matrix tri-factorization problem, which preserves the geometrical structures of data and feature manifold. Finally, it is worth mentioning [163], where the authors address the problem of multiple alternative clusterings, by introducing the first algorithm that generates multiple alternative co-clusterings at the same time. Even in this case, the matrix tri-factorization approach is considered and exploited to quantify row and column redundancy and enforce diversity among co-clustering solutions.

Further recent contributions are aimed at improving model-based text co-clustering by leveraging the **von Mises-Fisher (vMF)** mixture model and a bi-directional multiplicative regularization [3, 4] or enhancing bipartite graph partition-based co-clustering with a more flexible bipartite graph model [39]. Finally, the problem of correctly evaluating co-clusters has also been addressed recently by Robert et al. [142]. They adapt the the **Adjusted Rand Index (ARI)** introduced by Hubert and Arabie [86] to the co-clustering problem together with an efficient algorithm to compute it.

7.2 Open Problems and Promising Future Research Directions

As we said in the introduction, despite the continuous interest of the scientific community for co-clustering, in the past five years there has been a decreasing number of scientific contributions targeting it. We now provide some possible explanations for this apparent declining engagement of researchers in contributing to the co-clustering literature with novel approaches.

(1) **Scalability:** There are some challenging issues that make the computation of co-clustering solution unfeasible in very large data. For instance, the inherent time and space

complexity of eigenvalue decomposition, adopted by spectral methods, limits the application of such methods on very large data with many co-clusters. Another significant problem is that, differently from clustering, designing a distributed or parallel algorithm for a co-clustering objective function is challenging, since it is strictly connected to both partitions (or to the partitions on all modes, in case of tensor clustering). Hence, if, for instance, a horizontally partitioned schema is considered, then one can not simply aggregate local solutions computed in a parallel or distributed fashion, as the column partition is likely inconsistent among all nodes/threads participating in the computation without sharing other information, thus increasing the computational overhead and the risk of privacy leaks. Some solutions to the above-mentioned problems may come from recent advances in the stochastic optimization of co-representation learning algorithms [40].

(2) **Strong requirements:** Co-clustering is typically applied on very high-dimensional data where the input data matrix has approximately the same number of rows and columns ($n \sim m$) and, additionally, is very sparse. Consequently having a unique partition on the whole feature space is somehow too stringent a constraint. In this case, subspace clustering and biclustering offer a major flexibility, as multiple clusters or partitions are allowed on different subspaces. Indeed, a very recent approach tackles this issue by proposing a subspace co-clustering method [62]. Nonetheless, this limitation also affects one-side clustering algorithms, which, however, are permanently under the lens of the machine learning and statistics research communities. Moreover, another problem affecting co-clustering methods is that, like some one-mode clustering ones, they tend to converge towards highly skewed solutions with very unbalanced or even empty clusters [145].

(3) **Competing methods:** Although co-clustering has been proven effective in high-dimensional data, one-side clustering is still preferred, thanks to its good scalability properties and to the wide varierty of approaches, including graph-based ones and those based on matrix factorization and deep neural networks. Furthermore, in the past decade, correlation clustering [15, 100] has emerged as a useful method for partitioning high-dimensional data, with the additional interesting property of finding the number of clusters automatically.

Despite all these discouraging aspects, there are still some important open problems that the scientific community has not addressed or that have not received a significant attention yet. In the following, we discuss them by also reporting some preliminary attempts for solving them.

(1) **Federated co-clustering:** As federated learning has emerged as the *de facto* standard for distributed (and private) machine learning [94], many supervised and unsupervised federated algorithms have been proposed, mainly based on distributed execution of the **stochastic gradient descent (SGD)** algorithm. Clustering and matrix factorization have been the object of several contributions in this research field as well [99, 108, 109, 133, 166]. However, no federated framework for co-clustering has been proposed so far. As we already said, this could be in part due to the difficulty in designing a working aggregating schema for the objective functions of co-clustering algorithms, although some promising research direction could come from some recent advances in deep neural network architectures for co-clustering [40, 176], as they could lead to some natural federated extensions.

(2) **Privacy-preserving co-clustering:** Privacy is a major concern today, and many privacy-preserving machine learning algorithms have been proposed in the past 25 years [113]. It has been shown that even one-side clustering algorithms (such as k-means) may harm the privacy of data subjects [123, 159] and, for this reason, some privacy-preserving

clustering algorithms, also based on differential privacy [58], have been proposed [93, 125, 154]. Although the results of a co-clustering algorithm could provide further harm to individual privacy, represented by the column clustering solution, only a few works have addressed the problem of privacy-preserving co-clustering. They are mainly focused on distributed frameworks [81, 156], although co-clustering has been used as a means to generate differentially private synthetic data [21].

(3) **Number of parameters:** As we already pointed out in Section 5, most co-clustering approaches require a large number of hyper-parameters to be tuned, including the number of clusters for the rows and for the columns. For instance, a deep learning approach, such as Reference [171], requires the correct configuration of at least six input parameters to make it produce reliable results. Although some parameter-less approaches exist [18, 19, 141], major efforts could be taken for helping users decide the correct number of clusters per mode or make algorithms truly parameter-free and computationally efficient.

(4) **Alternative measures for co-clustering.** Most co-clustering objective functions rely on algebraic operations in the Euclidean space or association measures computed on the co-cluster contingency table, while, in the one-side clustering literature, many algorithms can be adapted to deal with virtually any definition of pairwise distance and similarity. Thanks to the recent advances in representation learning using deep neural networks, and the fact that, as already pointed out, some recent work on co-clustering based on deep learning has emerged, the next step would be identifying more complex relationships among row and column partitions, also involving alternative association measures. Interestingly, the notion of density-based co-clusters could also be significant but has never been investigated.

(5) **Generalization analysis.** When co-clustering is used in predictive tasks, it is crucial to assess its generalization performance. However, only a few works address the problem of analyzing the generalization error of co-clustering algorithms formally. Among others, the most notable example is Reference [148], where two predictive scenarios are considered: the prediction of the missing entries in data matrices (as in collaborative filtering) and the estimation of the joint probability distribution of row and column variables in co-occurrence matrices (as in word-document analysis). The authors analyze and derive generalization bounds under the PAC-Bayesian framework. Although many algorithms, especially those based on matrix factorization (see Section 4.2) include some regularization term in their objective functions, further advances on generalization error analysis for different co-clustering classes of methods would be beneficial for the development of the research field.

We recall that co-clustering can be reconsidered as an intrinsically interpretable clustering method in high-dimensional data and, as such, it meets recently introduced legal requirements concerning the explanation of decisions taken with the help of automated machine learning methods. Consequently, it is reasonable to expect a new research interest peak in the next few years.

8 Conclusion

In this article, we have surveyed the scientific literature about co-clustering, a popular unsupervised machine learning tool that deals with high-dimensional data. The goal of our survey is twofold: On the one hand, it is aimed at providing the uninformed reader with a practical overview of the main categories of methods and algorithms by also distinguishing them from similar approaches as subspace clustering and biclustering; on the other hand, it can help the expert reader with some insights on the limitations of existing approaches, the most recent trends, and possible promising future research directions. We have started our journey by presenting the main methods for matrices, but we have reviewed higher-order co-clustering methods for both N-way tensors and multi-view data as well.

As we have highlighted in our survey, it seems that, after a publication peak in 2019, in the past five years, the focus of the machine learning scientific community has shifted towards other methods with similar characteristics as those of co-clustering. Despite this, the general attraction for co-clustering in the overall scientific literature is increasing constantly. Hence, we think that it is worth investigating the open research issues on co-clustering topics. In this article, we have specifically discussed this, and we have given some suggestions by identifying federated co-clustering, privacy-preserving co-clustering, and more flexible co-clustering definitions among the most worthy future research tracks.

References

[1] Maryam Abdolali and Nicolas Gillis. 2021. Beyond linear subspace clustering: A comparative study of nonlinear manifold clustering algorithms. *Comput. Sci. Rev.* 42 (2021), 100435.

[2] Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. 2020. Ensemble block co-clustering: A unified framework for text data. In *Proceedings of the ACM CIKM*. 5–14.

[3] Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. 2021. Regularized bi-directional co-clustering. *Stat. Comput.* 31, 3 (2021), 32.

[4] Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. 2021. Regularized dual-PPMI co-clustering for text data. In *Proceedings of the ACM SIGIR*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). 2263–2267.

[5] Saeed Reza Aghabozorgi, Ali Seyed Shirkhorshidi, and Ying Wah Teh. 2015. Time-series clustering—A decade review. *Inf. Syst.* 53 (2015), 16–38.

[6] Melissa Ailem, François Role, and Mohamed Nadif. 2016. Graph modularity maximization as an effective method for co-clustering text data. *Knowl. Based Syst.* 109 (2016), 160–173.

[7] Melissa Ailem, François Role, and Mohamed Nadif. 2017. Model-based co-clustering for the effective handling of sparse data. *Pattern Recognit.* 72 (2017), 108–122.

[8] Melissa Ailem, François Role, and Mohamed Nadif. 2017. Sparse Poisson latent block model for document clustering. *IEEE Trans. Knowl. Data Eng.* 29, 7 (2017), 1563–1576.

[9] Kais Allab, Lazhar Labiod, and Mohamed Nadif. 2017. Multi-manifold matrix decomposition for data co-clustering. *Pattern Recognit.* 64 (2017), 386–398.

[10] Mohd Yousuf Ansari, Amir Ahmad, Shehroz S. Khan, Gopal Bhushan, and Mainuddin. 2020. Spatiotemporal clustering: A review. *Artif. Intell. Rev.* 53, 4 (2020), 2381–2423.

[11] Miguel Araujo, Pedro Manuel Pinto Ribeiro, and Christos Faloutsos. 2018. TensorCast: Forecasting time-evolving networks with contextual information. In *Proceedings of the IJCAI*, Jérôme Lang (Ed.). 5199–5203.

[12] Juhee Bae, Tove Helldin, Maria Riveiro, Slawomir Nowaczyk, Mohamed-Rafik Bouguelia, and Göran Falkman. 2020. Interactive clustering: A comprehensive review. *ACM Comput. Surv.* 53, 1 (2020), 1:1–1:39.

[13] Arindam Banerjee, Sugato Basu, and Srujana Merugu. 2007. Multi-way clustering on relation graphs. In *Proceedings of the SIAM SDM*. 145–156.

[14] Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S. Modha. 2007. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *J. Mach. Learn. Res.* 8 (2007), 1919–1986.

[15] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation clustering. *Mach. Learn.* 56, 1–3 (2004), 89–113.

[16] Andrea Baraldi and Palma Blonda. 1999. A survey of fuzzy clustering algorithms for pattern recognition. I. *IEEE Trans. Syst. Man Cybern. Part B* 29, 6 (1999), 778–785.

[17] Andrea Baraldi and Palma Blonda. 1999. A survey of fuzzy clustering algorithms for pattern recognition. II. *IEEE Trans. Syst. Man Cybern. Part B* 29, 6 (1999), 786–801.

[18] Elena Battaglia, Federico Peiretti, and Ruggero G. Pensa. 2024. Fast parameterless prototype-based co-clustering. *Mach. Learn.* 113, 4 (2024), 2153–2181.

[19] Elena Battaglia and Ruggero G. Pensa. 2023. A parameter-less algorithm for tensor co-clustering. *Mach. Learn.* 112, 2 (2023), 385–427.

[20] Florian Beil, Martin Ester, and Xiaowei Xu. 2002. Frequent term-based text clustering. In *Proceedings of the ACM SIGKDD*. 436–442.

[21] Tarek Benkhelif, Françoise Fessant, Fabrice Clérot, and Guillaume Raschia. 2017. Co-clustering for differentially private synthetic data generation. In *Proceedings of the ECML PKDD PAP Workshop*. 36–47.

[22] Christophe Biernacki, Julien Jacques, and Christine Keribin. 2023. A survey on model-based co-clustering: High dimension and estimation challenges. *J. Classif.* 40, 2 (2023), 332–381. DOI: https://doi.org/10.1007/S00357-023-09441-3

[23] H. H. Bock. 1980. Simultaneous clustering of objects and variables. In *Analyse de Données et Informatique*, R. Tomassone, M. Amirchanhy, and D. Néel (Eds.). INRIA, 187–203.

[24] Fred H. Borgen and David C. Barnett. 1987. Applying cluster analysis in counseling psychology research. *J. Counsel. Psychol.* 34, 4 (1987), 456.

[25] Léon Bottou and Olivier Bousquet. 2007. The tradeoffs of large scale learning. In *Proceedings of the NIPS*. 161–168.

[26] Rafika Boutalbi, Lazhar Labiod, and Mohamed Nadif. 2019. Co-clustering from tensor data. In *Proceedings of the PAKDD*, Vol. 11439. 370–383.

[27] Rafika Boutalbi, Lazhar Labiod, and Mohamed Nadif. 2019. Sparse tensor co-clustering as a tool for document categorization. In *Proceedings of the ACM SIGIR*. 1157–1160.

[28] Rafika Boutalbi, Lazhar Labiod, and Mohamed Nadif. 2021. Implicit consensus clustering from multiple graphs. *Data Min. Knowl. Discov.* 35, 6 (2021), 2313–2340.

[29] Rafika Boutalbi, Lazhar Labiod, and Mohamed Nadif. 2022. TensorClus: A Python library for tensor (co)-clustering. *Neurocomputing* 468 (2022), 464–468.

[30] Nicoletta Del Buono and Gianvito Pio. 2015. Non-negative matrix tri-factorization for co-clustering: An analysis of the block matrix. *Inf. Sci.* 301 (2015), 13–26.

[31] Jianghui Cai, Jing Hao, Haifeng Yang, Xujun Zhao, and Yuqing Yang. 2023. A review on semi-supervised clustering. *Inf. Sci.* 632 (2023), 164–200.

[32] Rui Cai, Lie Lu, and Alan Hanjalic. 2008. Co-clustering for auditory scene categorization. *IEEE Trans. Multim.* 10, 4 (2008), 596–606.

[33] Loïc Cerf, Jérémy Besson, Kim-Ngan Nguyen, and Jean-François Boulicaut. 2013. Closed and noise-tolerant patterns in n-ary relations. *Data Min. Knowl. Discov.* 26, 3 (2013), 574–619.

[34] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. 2009. Closed patterns meet *n*-ary relations. *ACM Trans. Knowl. Discov. Data* 3, 1 (2009), 3:1–3:36.

[35] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S. Modha, and Christos Faloutsos. 2004. Fully automatic cross-associations. In *Proceedings of the ACM SIGKDD*. 79–88.

[36] Guoqing Chao, Shiliang Sun, and Jinbo Bi. 2021. A survey on multiview clustering. *IEEE Trans. Artif. Intell.* 2, 2 (2021), 146–168.

[37] Malika Charrad, Yves Lechevallier, Mohamed Ben Ahmed, and Gilbert Saporta. 2009. Block clustering for web pages categorization. In *Proceedings of the IDEAL*. 260–267.

[38] Li Chen and Feng Wang. 2013. Preference-based clustering reviews for augmenting e-commerce recommendation. *Knowl. Based Syst.* 50 (2013), 44–59.

[39] Wei Chen, Hongjun Wang, Zhiguo Long, and Tianrui Li. 2023. Fast flexible bipartite graph model for co-clustering. *IEEE Trans. Knowl. Data Eng.* 35, 7 (2023), 6930–6940.

[40] Wei Chen, Hongjun Wang, Yinghui Zhang, Ping Deng, Zhipeng Luo, and Tianrui Li. 2024. T-distributed stochastic neighbor embedding for co-representation learning. *ACM Trans. Intell. Syst. Technol.* 15, 2, Article 23 (2024), 18 pages.

[41] Yufu Chen, Zhiqi Lei, Yanghui Rao, Haoran Xie, Fu Lee Wang, Jian Yin, and Qing Li. 2023. Parallel non-negative matrix tri-factorization for text data co-clustering. *IEEE Trans. Knowl. Data Eng.* 35, 5 (2023), 5132–5146.

[42] Yizong Cheng and George M. Church. 2000. Biclustering of expression data. In *Proceedings of the ISMB*. AAAI, 93–103.

[43] Eric C. Chi, Brian J. Gaines, Will Wei Sun, Hua Zhou, and Jian Yang. 2020. Provable convex co-clustering of tensors. *J. Mach. Learn. Res.* 21 (2020), 214:1–214:58.

[44] Hyuk Cho, Inderjit S. Dhillon, Yuqiang Guan, and Suvrit Sra. 2004. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the SIAM SDM*. 114–125.

[45] Andrej Copar, Marinka Zitnik, and Blaz Zupan. 2017. Scalable non-negative matrix tri-factorization. *BioData Min.* 10, 1 (2017), 41:1–41:16.

[46] Marco Corneli, Charles Bouveyron, and Pierre Latouche. 2020. Co-clustering of ordinal data via latent continuous random variables and not missing at random entries. *J. Comput. Graph. Stat.* 29, 4 (2020), 771–785.

[47] Gianni Costa, Giuseppe Manco, and Riccardo Ortale. 2008. A hierarchical model-based approach to co-clustering high-dimensional data. In *Proceedings of the ACM SAC*. 886–890.

[48] Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA.

[49] Zineb Dafir, Yasmine Lamari, and Said Chah Slaoui. 2021. A survey on parallel clustering algorithms for big data. *Artif. Intell. Rev.* 54, 4 (2021), 2411–2443.

[50] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2007. Co-clustering based classification for out-of-domain documents. In *Proceedings of the ACM SIGKDD*. ACM, 210–219.

[51] Jonathan de Andrade Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André Carlos Ponce de Leon Ferreira de Carvalho, and João Gama. 2013. Data stream clustering: A survey. *ACM Comput. Surv.* 46, 1 (2013), 13:1–13:31.

[52] Ping Deng, Tianrui Li, Hongjun Wang, Shi-Jinn Horng, Zeng Yu, and Xiaomin Wang. 2021. Tri-regularized nonnegative matrix tri-factorization for co-clustering. *Knowl. Based Syst.* 226 (2021), 107101.

[53] Inderjit S. Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the ACM SIGKDD*, Doheon Lee, Mario Schkolnick, Foster J. Provost, and Ramakrishnan Srikant (Eds.). 269–274.

[54] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. 2003. Information-theoretic co-clustering. In *Proceedings of the ACM SIGKDD.* 89–98.

[55] Chris H. Q. Ding and Xiaofeng He. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the SIAM SDM.* 606–610.

[56] Chris H. Q. Ding, Tao Li, Wei Peng, and Haesun Park. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the ACM SIGKDD.* 126–135.

[57] Shide Du, Zhanghui Liu, Zhaoliang Chen, Wenyuan Yang, and Shiping Wang. 2021. Differentiable bi-sparse multiview co-clustering. *IEEE Trans. Signal Process.* 69 (2021), 4623–4636.

[58] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (2014), 211–407.

[59] Beyza Ermis, Evrim Acar, and Ali Taylan Cemgil. 2015. Link prediction in heterogeneous data via generalized coupled tensor factorization. *Data Min. Knowl. Discov.* 29, 1 (2015), 203–236.

[60] Liang Feng, Qianchuan Zhao, and Cangqi Zhou. 2020. Improving performances of top-*N* recommendations with co-clustering method. *Expert Syst. Appl.* 143 (2020).

[61] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. 2022. Efficient and effective optimal transport-based biclustering. In *Proceedings of the NeurIPS.*

[62] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. 2024. Boosting subspace co-clustering via bilateral graph convolution. *IEEE Trans. Knowl. Data Eng.* 36, 3 (2024), 960–971.

[63] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. 2008. A survey of kernel and spectral methods for clustering. *Pattern Recognit.* 41, 1 (2008), 176–190.

[64] Christopher J. Fluke and Colin Jacobs. 2020. Surveying the reach and maturity of machine learning and artificial intelligence in astronomy. *Wiley Interdisc. Rev.: Data Min. Knowl. Discov.* 10, 2 (2020), e1349.

[65] Pedro A. Forero and Paul A. Baxley. 2020. Tucker-regularized tensor Bregman co-clustering. In *Proceedings of the EUSIPCO.* 1497–1501.

[66] Lele Fu, Pengfei Lin, Athanasios V. Vasilakos, and Shiping Wang. 2020. An overview of recent multi-view clustering. *Neurocomputing* 402 (2020), 148–161.

[67] Bin Gao, Tie-Yan Liu, Xin Zheng, QianSheng Cheng, and Wei-Ying Ma. 2005. Consistent bipartite graph copartitioning for star-structured high-order heterogeneous data co-clustering. In *Proceedings of the ACM SIGKDD.* 41–50.

[68] Thomas George and Srujana Merugu. 2005. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the IEEE ICDM.* 625–628.

[69] L. A. Goodman and W. H. Kruskal. 1954. Measures of association for cross classification. *J. Am. Stat. Assoc.* 49 (1954), 732–764.

[70] Gérard Govaert. 1995. Simultaneous clustering of rows and columns. *Contr. Cybern.* 24, 4 (1995), 437–458.

[71] Gérard Govaert and Mohamed Nadif. 2003. Clustering with block mixture models. *Pattern Recognit.* 36, 2 (2003), 463–473.

[72] Gérard Govaert and Mohamed Nadif. 2008. Block clustering with Bernoulli mixture models: Comparison of different approaches. *Comput. Stat. Data Anal.* 52, 6 (2008), 3233–3245.

[73] Gérard Govaert and Mohamed Nadif. 2010. Latent block model for contingency table. *Commun. Stat. - Theor. Meth.* 39, 3 (2010), 416–425.

[74] Gérard Govaert and Mohamed Nadif. 2013. *Co-clustering: Models, Algorithms and Applications.* John Wiley & Sons.

[75] Gérard Govaert and Mohamed Nadif. 2018. Mutual information, phi-squared and model-based co-clustering for contingency tables. *Adv. Data Anal. Classif.* 12, 3 (2018), 455–488.

[76] Quanquan Gu and Jie Zhou. 2009. Co-clustering on manifolds. In *Proceedings of the KDD.* 359–368.

[77] Richard A. Harshman. 1970. Foundation of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. *UCLA Work. Pap. Phonet.* 16 (1970), 1–84.

[78] John A. Hartigan. 1972. Direct clustering of a data matrix. *J. Am. Stat. Assoc.* 67, 337 (1972), 123–129.

[79] Jing He, Xin Li, Lejian Liao, and Mingzhong Wang. 2018. Inferring continuous latent preference on transition intervals for next point-of-interest recommendation. In *Proceedings of the ECML PKDD.* 741–756.

[80] Rui Henriques and Sara C. Madeira. 2019. Triclustering algorithms for three-dimensional data analysis: A comprehensive survey. *ACM Comput. Surv.* 51, 5 (2019), 95:1–95:43.

[81] Katsuhiro Honda, Shotaro Matsuzaki, Seiki Ubukata, and Akira Notsu. 2018. Privacy preserving collaborative fuzzy co-clustering of three-mode cooccurrence data. In *Proceedings of the MDAI*. 232–242.

[82] Min-Sung Hong and Jason J. Jung. 2018. Multi-sided recommendation based on social tensor factorization. *Inf. Sci.* 447 (2018), 140–156.

[83] Eduardo R. Hruschka, Ricardo José Gabrielli Barreto Campello, Alex Alves Freitas, and André Carlos Ponce de Leon Ferreira de Carvalho. 2009. A survey of evolutionary algorithms for clustering. *IEEE Trans. Syst. Man Cybern. Part C* 39, 2 (2009), 133–155.

[84] Shudong Huang, Hongjun Wang, Dingcheng Li, Yan Yang, and Tianrui Li. 2015. Spectral co-clustering ensemble. *Knowl. Based Syst.* 84 (2015), 46–55.

[85] Shudong Huang, Zenglin Xu, Ivor W. Tsang, and Zhao Kang. 2020. Auto-weighted multi-view co-clustering with bipartite graphs. *Inf. Sci.* 512 (2020), 18–30.

[86] Lawrenc Hubert and Phipps Arabie. 1985. Comparing partitions. *J. Classif.* 2, 1 (1985), 193–218.

[87] Syed Fawad Hussain, Khadija Khan, and Rashad M. Jillani. 2022. Weighted multi-view co-clustering (WMVCC) for sparse data. *Appl. Intell.* 52, 1 (2022), 398–416.

[88] Dino Ienco, Céline Robardet, Ruggero G. Pensa, and Rosa Meo. 2013. Parameter-less co-clustering for star-structured heterogeneous data. *Data Min. Knowl. Discov.* 26, 2 (2013), 217–254.

[89] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Heming Jia. 2023. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Inf. Sci.* 622 (2023), 178–210.

[90] Julien Jacques and Christophe Biernacki. 2018. Model-based co-clustering for ordinal data. *Comput. Stat. Data Anal.* 123 (2018), 101–115.

[91] Anil K. Jain and Richard C. Dubes. 1988. *Algorithms for Clustering Data.* Prentice-Hall.

[92] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. 1999. Data clustering: A review. *ACM Comput. Surv.* 31, 3 (1999), 264–323.

[93] Somesh Jha, Luis Kruger, and Patrick McDaniel. 2005. Privacy preserving clustering. In *Proceedings of the ESORICS*. Springer, 397–417.

[94] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* 14, 1–2 (2021), 1–210.

[95] Christine Keribin, Vincent Brault, Gilles Celeux, and Gérard Govaert. 2015. Estimation and selection for the latent block model on categorical data. *Stat. Comput.* 25, 6 (2015), 1201–1216.

[96] Margret Keuper, Siyu Tang, Bjoern Andres, Thomas Brox, and Bernt Schiele. 2020. Motion segmentation & multiple object tracking by correlation co-clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 1 (2020), 140–153.

[97] Yuval Kluger, Ronen Basri, Joseph T. Chang, and Mark Gerstein. 2003. Spectral biclustering of microarray cancer data: Co-clustering genes and conditions. *Genome Rese.* 13 (2003), 703–716.

[98] Tamara G. Kolda and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM Rev.* 51, 3 (2009), 455–500.

[99] Aashish Kolluri, Teodora Baluta, and Prateek Saxena. 2021. Private hierarchical clustering in federated networks. In *Proceedings of the ACM SIGSAC CCS*, Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi (Eds.). 2342–2360.

[100] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. 2009. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data* 3, 1 (2009), 1:1–1:58.

[101] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. 2012. Subspace clustering. *WIREs Data Mining Knowl. Discov.* 2, 4 (2012), 351–364.

[102] Lazhar Labiod and Mohamed Nadif. 2011. Co-clustering for binary and categorical data with maximum modularity. In *Proceedings of the IEEE ICDM*. 1140–1145.

[103] Charlotte Laclau and Mohamed Nadif. 2017. Diagonal latent block model for binary data. *Stat. Comput.* 27, 5 (2017), 1145–1163.

[104] Charlotte Laclau, Ievgen Redko, Basarab Matei, Younès Bennani, and Vincent Brault. 2017. Co-clustering through optimal transport. In *Proceedings of the ICML*, Vol. 70. 1955–1964.

[105] Pedro Larrañaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Iñaki Inza, José Antonio Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez Martínez, and Victor Robles. 2006. Machine learning in bioinformatics. *Brief. Bioinform.* 7, 1 (2006), 86–112.

[106] Ping Li, Jiajun Bu, Chun Chen, Zhanying He, and Deng Cai. 2013. Relational multimanifold coclustering. *IEEE Trans. Cybern.* 43, 6 (2013), 1871–1881.

[107] Xiangli Li, Xiyan Lu, and Xuezhen Fan. 2022. Semi-supervised sparse neighbor constrained co-clustering with dissimilarity and similarity regularization. *Eng. Appl. Artif. Intell.* 114 (2022), 104989.

[108] Zitao Li, Bolin Ding, Ce Zhang, Ninghui Li, and Jingren Zhou. 2021. Federated matrix factorization with privacy guarantee. *Proc. VLDB Endow.* 15, 4 (2021), 900–913.

[109] Zitao Li, Tianhao Wang, and Ninghui Li. 2023. Differentially private vertical federated clustering. *Proc. VLDB Endow.* 16, 6 (2023), 1277–1290.

[110] Tingting Liang, Liang Chen, Haochao Ying, and Jian Wu. 2014. Co-clustering WSDL documents to bootstrap service discovery. In *Proceedings of the IEEE SOCA*. 215–222.

[111] T. Warren Liao. 2005. Clustering of time series data—A survey. *Pattern Recognit.* 38, 11 (2005), 1857–1874.

[112] Renjie Lin, Shiping Wang, and Wenzhong Guo. 2019. An overview of co-clustering via matrix factorization. *IEEE Access* 7 (2019), 33481–33493.

[113] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. 2022. When machine learning meets privacy: A survey and outlook. *ACM Comput. Surv.* 54, 2 (2022), 31:1–31:36.

[114] Na Liu, Fei Chen, and Mingyu Lu. 2013. Spectral co-clustering documents and words using fuzzy K-harmonic means. *Int. J. Mach. Learn. Cybern.* 4, 1 (2013), 75–83.

[115] Bo Long, Zhongfei (Mark) Zhang, and Philip S. Yu. 2005. Co-clustering by block value decomposition. In *Proceedings of the ACM SIGKDD*. 635–640.

[116] Sara C. Madeira and Arlindo L. Oliveira. 2004. Biclustering algorithms for biological data analysis: A survey. *IEEE ACM Trans. Comput. Biol. Bioinform.* 1, 1 (2004), 24–45.

[117] Vichi Maurizio. 2001. Double k-means clustering for simultaneous classification of objects and variables. In *Advances in Classification and Data Analysis*. Springer Berlin, 43–52.

[118] Iven Mechelen, Hans-Hermann Bock, and Paul De Boeck. 2004. Two-mode clustering methods: A structured overview. *Stat. Meth. Med. Res.* 13 (11 2004), 363–94.

[119] Mohamed Nadif and Gérard Govaert. 2010. Model-based co-clustering for continuous data. In *Proceedings of the IEEE ICMLA*. 175–180.

[120] Mark E. J. Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Phys. Rev. E* 69, 2 (2004), 026113.

[121] Kim-Ngan Nguyen, Loïc Cerf, Marc Plantevit, and Jean-François Boulicaut. 2011. Multidimensional association rules in Boolean tensors. In *Proceedings of the SIAM SDM*. 570–581.

[122] Feiping Nie, Shaojun Shi, and Xuelong Li. 2020. Auto-weighted multi-view co-clustering via fast matrix factorization. *Pattern Recognit.* 102 (2020), 107207.

[123] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the STOC*. 75–84.

[124] Marta D. M. Noronha, Rui Henriques, Sara C. Madeira, and Luis E. Zárate. 2022. Impact of metrics on biclustering solution and quality: A review. *Pattern Recognit.* 127 (2022), 108612.

[125] Stanley R. M. Oliveira and Osmar R. Zaiane. 2010. Privacy preserving clustering by data transformation. *J. Inf. Data Manag.* 1, 1 (2010), 37–37.

[126] Divya Pandove, Shivani Goel, and Rinkle Rani. 2018. Systematic review of clustering high-dimensional and large datasets. *ACM Trans. Knowl. Discov. Data* 12, 2 (2018), 16:1–16:68.

[127] Spiros Papadimitriou and Jimeng Sun. 2008. DisCo: Distributed co-clustering with Map-Reduce: A case study towards petabyte-scale end-to-end mining. In *Proceedings of the IEEE ICDM*. 512–521.

[128] Evangelos E. Papalexakis, Alex Beutel, and Peter Steenkiste. 2018. Network anomaly detection using co-clustering. In *Encyclopedia of Social Network Analysis and Mining, 2nd Edition*. Springer.

[129] Evangelos E. Papalexakis and A. Seza Doğruöz. 2015. Understanding multilingual social networks in online immigrant communities. In *Proceedings of the WWW*. 865–870.

[130] Evangelos E. Papalexakis, Nicholas D. Sidiropoulos, and Rasmus Bro. 2013. From k-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors. *IEEE Trans. Signal Process.* 61, 2 (2013), 493–506.

[131] Lance Parsons, Ehtesham Haque, and Huan Liu. 2004. Subspace clustering for high dimensional data: A review. *SIGKDD Explor.* 6, 1 (2004), 90–105.

[132] Karl Pearson. 1900. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philos. Mag.* 5 (1900), 157–175.

[133] Witold Pedrycz. 2022. Federated FCM: Clustering under privacy requirements. *IEEE Trans. Fuzzy Syst.* 30, 8 (2022), 3384–3388.

[134] Ruggero G. Pensa and Jean-François Boulicaut. 2008. Constrained co-clustering of gene expression data. In *Proceedings of the SIAM SDM.* 25–36.

[135] Ruggero G. Pensa, Dino Ienco, and Rosa Meo. 2014. Hierarchical co-clustering: Off-line and incremental approaches. *Data Min. Knowl. Discov.* 28, 1 (2014), 31–64.

[136] Beatriz Pontes, Raúl Giráldez, and Jesús S. Aguilar-Ruiz. 2015. Biclustering on expression data: A review. *J. Biomed. Inform.* 57 (2015), 163–180.

[137] Guoping Qiu. 2004. Image and feature co-clustering. In *Proceedings of the ICPR.* 991–994.

[138] Manjeet Rege, Ming Dong, and Farshad Fotouhi. 2006. Co-clustering documents and words using bipartite isoperimetric graph partitioning. In *Proceedings of the IEEE ICDM.* 532–541.

[139] Paul Riverain, Simon Fossier, and Mohamed Nadif. 2022. Semi-supervised latent block model with pairwise constraints. *Mach. Learn.* 111, 5 (2022), 1739–1764.

[140] Paul Riverain, Simon Fossier, and Mohamed Nadif. 2023. Poisson degree corrected dynamic stochastic block model. *Adv. Data Anal. Classif.* 17, 1 (2023), 135–162.

[141] Céline Robardet and Fabien Feschet. 2001. Efficient local search in conceptual clustering. In *Proceedings of the DS (Lecture Notes in Computer Science)*, Vol. 2226. 323–335.

[142] Valerie Robert, Yann Vasseur, and Vincent Brault. 2021. Comparing high-dimensional partitions with the co-clustering adjusted rand index. *J. Classif.* 38, 1 (2021), 158–186.

[143] Roberto Rocci and Maurizio Vichi. 2008. Two-mode multi-partitioning. *Comput. Stat. Data Anal.* 52, 4 (2008), 1984–2003.

[144] Aghiles Salah, Melissa Ailem, and Mohamed Nadif. 2018. Word co-occurrence regularized non-negative matrix tri-factorization for text data co-clustering. In *Proceedings of the AAAI.* 3992–3999.

[145] Aghiles Salah and Mohamed Nadif. 2017. Model-based Von Mises-Fisher co-clustering with a conscience. In *Proceedings of the SIAM SDM.* 246–254.

[146] Aghiles Salah and Mohamed Nadif. 2019. Directional co-clustering. *Adv. Data Anal. Classif.* 13, 3 (2019), 591–620.

[147] Aghiles Salah, Nicoleta Rogovschi, and Mohamed Nadif. 2016. Model-based co-clustering for high dimensional sparse data. In *Proceedings of the AISTATS*, Vol. 51. 866–874.

[148] Yevgeny Seldin and Naftali Tishby. 2010. PAC-Bayesian analysis of co-clustering and beyond. *J. Mach. Learn. Res.* 11 (2010), 3595–3646.

[149] Margot Selosse, Julien Jacques, and Christophe Biernacki. 2020. Model-based co-clustering for mixed type data. *Comput. Stat. Data Anal.* 144 (2020), 106866.

[150] Amnon Shashua and Tamir Hazan. 2005. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the ICML*, Vol. 119. 792–799.

[151] Kelvin Sim, Vivekanand Gopalkrishnan, Arthur Zimek, and Gao Cong. 2013. A survey on enhanced subspace clustering. *Data Min. Knowl. Discov.* 26, 2 (2013), 332–397.

[152] Yosra Ben Slimen, Sylvain Allio, and Julien Jacques. 2018. Model-based co-clustering for functional data. *Neurocomputing* 291 (2018), 97–108.

[153] Yangqiu Song, Shimei Pan, Shixia Liu, Furu Wei, Michelle X. Zhou, and Weihong Qian. 2010. Constrained coclustering for textual documents. In *Proceedings of the AAAI.* 581–586.

[154] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, Min Lyu, and Hongxia Jin. 2017. Differentially private k-means clustering and a hybrid approach to private optimization. *ACM Trans. Privac. Secur.* 20, 4 (2017), 1–33.

[155] Qi Tan, Pei Yang, and Jingrui He. 2018. Feature co-shrinking for co-clustering. *Pattern Recognit.* 77 (2018), 12–19.

[156] Daiji Tanaka, Toshiya Oda, Katsuhiro Honda, and Akira Notsu. 2014. Privacy preserving fuzzy co-clustering with distributed cooccurrence matrices. In *Proceedings of the SCIS/ISIS.* 700–705.

[157] Jiayi Tang and Zhong Wan. 2021. Orthogonal dual graph-regularized nonnegative matrix factorization for co-clustering. *J. Sci. Comput.* 87, 3 (2021), 66.

[158] Ledyard R. Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31 (1966), 279–311.

[159] Jaideep Vaidya, Yu Michael Zhu, and Christopher W. Clifton. 2006. *Privacy and Data Mining*. Springer.

[160] Rene Vidal. 2011. Subspace clustering. *IEEE Signal Process. Mag.* 28, 2 (2011), 52–68.

[161] Ulrike von Luxburg. 2007. A tutorial on spectral clustering. *Stat. Comput.* 17, 4 (2007), 395–416.

[162] Hua Wang, Feiping Nie, Heng Huang, and Fillia Makedon. 2011. Fast nonnegative matrix tri-factorization for large-scale data co-clustering. In *Proceedings of the IJCAI*, Toby Walsh (Ed.). 1553–1558.

[163] Jun Wang, Xing Wang, Guoxian Yu, Carlotta Domeniconi, Zhiwen Yu, and Zili Zhang. 2021. Discovering multiple co-clusterings with matrix factorization. *IEEE Trans. Cybern.* 51, 7 (2021), 3576–3587.

[164] Miaoyan Wang and Yuchen Zeng. 2019. Multiway clustering via tensor block models. In *Proceedings of the NeurIPS.* 713–723.

[165] Pu Wang, Carlotta Domeniconi, and Jian Hu. 2008. Using Wikipedia for co-clustering based cross-domain text classification. In *Proceedings of the IEEE ICDM*. 1085–1090.

[166] Shuai Wang and Tsung-Hui Chang. 2022. Federated matrix factorization: Algorithm design and application to data clustering. *IEEE Trans. Signal Process.* 70 (2022), 1625–1640.

[167] Yan Wang and Xiaoke Ma. 2021. Joint nonnegative matrix factorization and network embedding for graph co-clustering. *Neurocomputing* 462 (2021), 453–465.

[168] Yu-Xiong Wang and Yu-Jin Zhang. 2013. Nonnegative matrix factorization: A comprehensive review. *IEEE Trans. Knowl. Data Eng.* 25, 6 (2013), 1336–1353.

[169] Tao Wu, Austin R. Benson, and David F. Gleich. 2016. General tensor spectral co-clustering for higher-order data. In *Proceedings of the NIPS*. 2559–2567.

[170] Juan Xie, Anjun Ma, Anne Fennell, Qin Ma, and Jing Zhao. 2019. It is time to apply biclustering: A comprehensive review of biclustering applications in biological and biomedical data. *Brief. Bioinform.* 20, 4 (2019), 1450–1465.

[171] Dongkuan Xu, Wei Cheng, Bo Zong, Jingchao Ni, Dongjin Song, Wenchao Yu, Yuncong Chen, Haifeng Chen, and Xiang Zhang. 2019. Deep co-clustering. In *Proceedings of the SIAM SDM*. 414–422.

[172] Peng Xu, Zhaohong Deng, Kup-Sze Choi, Longbing Cao, and Shitong Wang. 2019. Multi-view information-theoretic co-clustering for co-occurrence data. In *Proceedings of the AAAI*. 379–386.

[173] Rui Xu and Donald C. Wunsch II. 2005. Survey of clustering algorithms. *IEEE Trans. Neural Netw.* 16, 3 (2005), 645–678.

[174] Wei Xu, Xin Liu, and Yihong Gong. 2003. Document clustering based on non-negative matrix factorization. In *Proceedings of the ACM SIGIR*. 267–273.

[175] Xiaoqiang Yan, Shizhe Hu, Yiqiao Mao, Yangdong Ye, and Hui Yu. 2021. Deep multi-view learning methods: A review. *Neurocomputing* 448 (2021), 106–129.

[176] Tianchi Yang, Cheng Yang, Luhao Zhang, Chuan Shi, Maodi Hu, Huaijun Liu, Tao Li, and Dong Wang. 2022. Co-clustering interactions via attentive hypergraph neural network. In *Proceedings of the SIGIR*. 859–869.

[177] Wu Yang, Guowei Shen, Wei Wang, Liangyi Gong, Miao Yu, and Guozhong Dong. 2015. Anomaly detection in microblogging via co-clustering. *J. Comput. Sci. Technol.* 30, 5 (2015), 1097–1108.

[178] Jiho Yoo and Seungjin Choi. 2010. Orthogonal nonnegative matrix tri-factorization for co-clustering: Multiplicative updates on Stiefel manifolds. *Inf. Process. Manag.* 46, 5 (2010), 559–570.

[179] Ke Yu, Lifang He, Philip S. Yu, Wenkai Zhang, and Yue Liu. 2019. Coupled tensor decomposition for user clustering in mobile internet traffic interaction pattern. *IEEE Access* 7 (2019), 18113–18124.

[180] Xianxue Yu, Guoxian Yu, Jun Wang, and Carlotta Domeniconi. 2021. Co-clustering ensembles based on multiple relevance measures. *IEEE Trans. Knowl. Data Eng.* 33, 4 (2021), 1389–1400.

[181] Pengcheng Zeng and Zhixiang Lin. 2021. coupleCoC+: An information-theoretic co-clustering-based transfer learning framework for the integrative analysis of single-cell genomic data. *PLoS Comput. Biol.* 17, 6 (2021), e1009064.

[182] Rui Zhang, Feiping Nie, Xuelong Li, and Xian Wei. 2019. Feature selection with multi-view data: A survey. *Inf. Fusion* 50 (2019), 158–167.

[183] Tong Zhang and Gene H. Golub. 2001. Rank-one approximation to high order tensors. *SIAM J. Matrix Anal. Applic.* 23, 2 (2001), 534–550.

[184] Zhongyuan Zhang, Tao Li, and Chris H. Q. Ding. 2013. Non-negative tri-factor tensor decomposition with applications. *Knowl. Inf. Syst.* 34, 2 (2013), 243–265.

[185] Lizhuang Zhao and Mohammed Javeed Zaki. 2005. TriCluster: An effective algorithm for mining coherent clusters in 3D microarray data. In *Proceedings of the ACM SIGMOD*. 694–705.

[186] Qingbiao Zhou, Guangdong Xu, and Yu Zong. 2009. Web Co-clustering of usage network using tensor decomposition. In *Proceedings of the IEEE/WIC/ACM WI/IAT Workshop ECBS*. IEEE Computer Society, 311–314.

[187] Yada Zhu and Jingrui He. 2016. Co-clustering structural temporal data with applications to semiconductor manufacturing. *ACM Trans. Knowl. Discov. Data* 10, 4 (2016), 43:1–43:18.

[188] Alaettin Zubaroglu and Volkan Atalay. 2021. Data stream clustering: A review. *Artif. Intell. Rev.* 54, 2 (2021), 1201–1236.

Received 5 June 2023; revised 16 September 2024; accepted 1 October 2024