Differentially Private Associative Co-clustering

Elena Battaglia* Ruggero G. Pensa*

Abstract

Co-clustering is a useful tool that extracts summary information from a data matrix in terms of row and column clusters, and gives a succinct representation of the data. However, if the matrix contains data about individuals, such representations could leak their privacy-sensitive information. In terms of privacy disclosure, co-clustering is even more harmful than clustering, because of the additional information carried by the column partition. However, to the best of our knowledge, the problem of privacy-preserving co-clustering has never been studied. To fill this gap, we consider a recent co-clustering algorithm, based on a de-normalized version of the Goodman-Kruskal's τ association measure, which has a good property from a differential privacy perspective, and is supposed not to consume an excessive amount of privacy budget. This leads to a privacy-preserving co-clustering algorithm that satisfies the definition of differential privacy while providing good partitioning solutions. Our algorithm is based on a prototype-based optimization strategy that makes it fast and actionable in realistic privacy-preserving data management and analysis scenarios, as shown by our extensive experimental validation.

1 Introduction

Clustering is by far one of the most popular machine learning tasks for its ability of extracting useful summaries of the data in the preliminary phases of a knowledge discovery process. When dealing with high-dimensional matrices, clustering is often superseded by co-clustering, which consists in processing the two modes (the rows and the columns) of the input matrix simultaneously [6, 11]. However, co-clustering could potentially harm the privacy of the individuals whose data are represented by the matrix. In fact, it has been shown that disclosing the results of standard clustering algorithms (such as k-means) may put the privacy of data subjects at risk [24, 30]. For this reason, many privacy-preserving clustering methods have been proposed [19,25,29]. Some privacy-preserving versions of k-means, in particular, have shown good results in terms of accuracy while satisfying strong theoretical privacy guarantees based on the definition of differential privacy, almost universally considered — although not without criticism [7] — as the *de facto* standard in terms of private data management and analysis. Intuitively, the additional information provided by column clusters in co-clustering results should make even more simple, for a malicious or adversarial user, to attack data subjects' privacy. Surprisingly, although some algorithms performing differentially private clustering of high-dimensional data do exist [1, 28], to the best of our knowledge, no privacy preserving co-clustering algorithm has been proposed so far.

This can be explained by the fact that, while in one-mode clustering few iterations turn out to be sufficient in standard applications [29], a co-clustering algorithm typically requires a higher number of iterations to get to a (sub)optimal solution, especially on large high dimensional data [3, 6, 31]. Indeed, since, during each iteration, a differentially private optimization strategy would consume part of the privacy budget ε, as stated by the composition theorem [20], privacy-preserving algorithms based on k-Means typically set a rather low upper bound to the maximum number of iterations. This would prevent the achievement of a reasonably high-quality co-clustering solution. However, recently, a fast co-clustering algorithm has been presented that has an interesting property, i.e., it very rapidly converges towards a solution consisting of an *a priori* unspecified number of clusters [2]. To do that, it optimizes a denormalized variant of a well-known association measure adopted in many matrix and tensor co-clustering algorithms, namely the Goodman-Kruskal's τ association measure [17].

In this paper, we propose a differentially private co-clustering algorithm, DP-τCC, that, for a given dataset, returns a solution consisting of a privacy-preserving co-clustering contingency matrix (indicating the strength of the link between each row cluster and every column cluster) and a differentially-private column-cluster mapping that can be used to infer the cluster assignment of every object, together with its description in terms of feature clusters. Both outcomes can be safely released and used for any subsequent task, such as product recommendation or user profiling and personalization. We prove the theoretical privacy guarantees of our method and show experimentally that DP-τCC provides good results even for relatively

*University of Turin, Italy, ruggero.pensa@unito.it

small privacy budgets and outperforms state-of-the-art differentially private methods for high-dimensional data clustering.

2 Related Work

Although co-clustering has been used to generate synthetic data in a differentially-private way [4], the problem of privacy-preserving co-clustering has not been studied yet. Indeed, differentially private clustering is a thriving field, even though almost all the proposed algorithms are private versions of k-means. The first ever algorithm for DP-k-means is DPLloyd [8], which modifies the Lloyd algorithm for k-means clustering injecting noise via the Laplace mechanism in the centroids computation step. An improved algorithm is given in [29], while [18, 24, 33] exploit the Sample and Aggregate Framework [24] to inject the noise on the output. Other approaches adopt input perturbation [16, 28, 29] or improve the clustering results by reducing the noise introduced by the randomized mechanisms [15, 23].

A method explicitly designed to perform private k-means on high dimensional datasets is introduced in [1]. Its idea is to project the data in a low-dimensional space using the Johnson-Lindenstrauss transform; a set of candidate prototypes are generated in a lower dimensional space and k centroids are selected. Finally, the centroids are recovered in the original high-dimensional space. The accuracy of the method is improved in [28], using approximate differential privacy.

Another class of methods that cope with high-dimensional data are those based on matrix factorization, although they are mostly used for privacy-preserving recommendation in distributed scenarios. They can be broadly divided into two groups. *Private ALS* methods, such as [5], use the Alternating Least Squares optimization and exploit the output perturbation for the private empirical risk minimization method [9] to inject the noise in the computation. *Private Gradient Descent* methods [5, 21, 32, 34] introduce the distortion in the gradient descent; this is the most common approach and the different algorithms vary in the mechanism used to add the noise and in the variant of the optimization algorithm. A more recent approach performs (ε,δ)-differentially private NMF by also considering the presence of outliers [26]. However, while the basis matrix is computed privately, the coefficient matrix is updated by accessing the data without consuming the privacy budget. Furthermore, the initialization is performed by applying randomized SVD on the original data. In our approach, instead, all steps are truly ε-differentially private.

It is worth noting that none of the aforementioned method performs matrix tri-factorization, which provides factor matrices for both rows and columns of the input data matrix, thus providing a natural decomposition method for co-clustering [10, 27].

3 Background and Motivation

In this section we introduce some basic notions on differential privacy and co-clustering that are required to understand how our privacy-preserving co-clustering algorithm works. Moreover, we provide an example that motivates the need for computing co-clustering results in a differentially private way.

3.1 Differential Privacy Differential privacy (DP) has gained great popularity since its introduction in 2006 [13]. Intuitively, it requires the output of a query to be approximately the same if any single tuple is added or removed from a dataset. In this way, an attacker is not able to understand whether a specific individual is included in the dataset based solely on the query outcome. This goal is obtained by injecting a controlled amount of randomness in the computation of the query result. The formal definition of differential privacy is the following:

Definition 1. (ε-differential privacy) *Let $\mathcal{M} : \Omega \longrightarrow \mathcal{R}$ be a stochastic function and $\varepsilon > 0$ a real number. We say that $\mathcal{M}$ preserves ε-differential privacy if, for any pairs of neighboring datasets $D, D' \in \Omega$ and $\forall r \in \mathcal{R}$,*

$$e^{-\varepsilon} \leq \frac{P(\mathcal{M}(D) = r)}{P(\mathcal{M}(D') = r)} \leq e^{\varepsilon}.$$

Two datasets are said to be *neighboring* if they only differ in one record. The parameter ε (also called *privacy budget*) controls the level of privacy guaranteed. Lower values of ε correspond to stronger privacy, since when $\varepsilon \to 0$, we have $e^{\varepsilon} \to 1$. In this case, the likelihood that the same outcome r comes from dataset D or from dataset D' is almost the same.

Usually, some randomization mechanism must be applied to transform the non-private function in a differentially private one. In this paper, we will use the Laplace mechanisms [13] and an improved version of the Exponential mechanisms [12] that increases the accuracy of the traditional Exponential mechanism [22] reducing the magnitude of the stochastic noise. The former can be applied to compute the result of a numeric function in a differentially private fashion; the latter is typically used to privately choose a result that maximizes a utility function whose outcome depends on the private dataset D. The amount of randomness injected in the computation by both mechanisms can be calibrated by looking at the sensitivity of the function (or utility function) they consider.

DEFINITION 2. (SENSITIVITY) *Let $q : \Omega \longrightarrow \mathbb{R}^d$ be a function. The sensitivity $\Delta(q)$ is the maximal variation of function q when computed on two neighboring datasets and is defined as $\Delta(q) = \max_{D \sim D'} ||q(D) - q(D')||_1$.*

DEFINITION 3. (LAPLACE MECHANISM) *Let $q : \Omega \longrightarrow \mathbb{R}^d$ be a function. The* Laplace mechanism *is $\mathcal{M}(D) = q(D) + (X_1, \ldots, X_d)$, where $X_1, \ldots, X_k$ are i.i.d. random variables drawn from a Laplace distribution with parameters $(0, \Delta(q)/\varepsilon)$.*

DEFINITION 4. (EXPONENTIAL MECHANISM [12]) *Let $q : \Omega \longrightarrow \mathcal{R}$ be the function that returns, among all possible values in $\mathcal{R}$, the one that maximizes a utility function $u : \Omega \times \mathcal{R} \longrightarrow \mathbb{R}$. The* Exponential mechanism *$\mathcal{M}(D)$ returns a value of $\mathcal{R}$ with probability proportional to $\exp(\varepsilon \cdot u(D, r)/\Delta(u))$, where*

$$\Delta(u) = \sup_{D \sim D'} \Big(\max_r \big(u(D, r) - u(D', r)\big) - \min_r \big(u(D, r) - u(D', r)\big) \Big).$$

It is proved that both mechanisms preserve ε-differential privacy [12, 14]. Moreover, [14] also introduces several composition theorems. More specifically, the *sequential composition* and the *parallel composition* theorems state that the composition of several differentially private functions is differentially private as well, and give a way to compute the overall privacy level of the computation. On the other hand, The *post-processing* theorem says that when an outcome r is computed in a differentially private way, any following transformation of this result is still differentially private and there is no need to allocate part of the privacy budget for it.

3.2 Fast co-clustering with de-normalized τ

Fast-τCC [2] is a recent co-clustering algorithm that has good convergence properties and is also able to identify a congruent number of clusters on rows and columns, starting from an initial overestimation. It is formalized as follows.

Given a data matrix $\mathbf{A} = (a_{ij}) \in \mathbb{R}_+^{n \times m}$, a co-clustering of $\mathbf{A}$ is a pair $(\mathcal{R}, \mathcal{C})$, where $\mathcal{R}$ is a partition of the rows and $\mathcal{C}$ a partition of the columns of the matrix. The objective function of Fast-τCC is derived from the Goodman and Kruskal's τ [17], and can be defined as an association index that estimates the strength of the link between two partitions $\mathcal{R}$ and $\mathcal{C}$ according to the reduction of the error in predicting $\mathcal{R}$ (resp. $\mathcal{C}$) knowing $\mathcal{C}$ (resp. $\mathcal{R}$):

$$\hat{\tau}_{R|C} = e_R - \mathbb{E}[e_{R|C}] \tag{3.1}$$

where e_R is the error in predicting $\mathcal{R}$, and $\mathbb{E}[e_{R|C}]$ is the expected value of the error in predicting $\mathcal{R}$ when $\mathcal{C}$ is known. Coming back to the input data matrix $\mathbf{A}$, $\hat{\tau}_{R|C}$ can be computed as

$$\hat{\tau}_{R|C}(\mathcal{R}, \mathcal{C}) = \sum_{k=1}^{|\mathcal{R}|} \sum_{l=1}^{|\mathcal{C}|} \frac{t_{kl}^2}{T \cdot t_{\cdot l}} - \sum_{k=1}^{|\mathcal{R}|} \frac{t_{k\cdot}^2}{T^2} \tag{3.2}$$

where $\mathbf{T} = (t_{kl})$ is the contingency table associated to the co-clustering $(\mathcal{R}, \mathcal{C})$, where $\mathcal{R} = (\mathcal{R}_1, \ldots, \mathcal{R}_K)$ and $\mathcal{C} = (\mathcal{C}_1, \ldots, \mathcal{C}_L)$, i.e. $t_{kl} = \sum_{i \in \mathcal{R}_k} \sum_{j \in \mathcal{C}_l} a_{ij}$, for $k = 1, \ldots, K$ and $l = 1, \ldots, L$. Following this notation, $t_{k\cdot} = \sum_{l=1}^{L} t_{kl}$, $t_{\cdot l} = \sum_{k=1}^{K} t_{kl}$ and $T = \sum_{k=1}^{K} \sum_{l=1}^{L} t_{kl}$.

Analogously, the association of the column clustering $\mathcal{C}$ to the row clustering $\mathcal{R}$ can be evaluated through the function $\hat{\tau}_{C|R}(\mathcal{R}, \mathcal{C})$. Since $\hat{\tau}$ is not symmetric, the best co-clustering solutions are those that simultaneously maximize $\hat{\tau}_{R|C}$ and $\hat{\tau}_{C|R}$.

In the following, we briefly summarize the main steps of the algorithm implementing Fast-τCC. First, we must introduce two matrices $\mathbf{P} = (p_{ij})$ and $\mathbf{Q} = (q_{kl})$, with $p_{ij} = \frac{a_{ij}}{A}$ and $q_{kl} = \frac{t_{kl}}{A} = \sum_{i \in \mathcal{R}_k} \sum_{j \in \mathcal{C}_l} p_{ij}$, where A denotes the sum of all the entries of $\mathbf{A}$ (hence, $A = T$). We also introduce the row cluster incidence matrix $\mathbf{R} = r_{ik}$ and $\mathbf{C} = c_{jl}$, with $r_{ik} = 1$ if row i is in row cluster $\mathcal{R}_k$ ($r_{ik} = 0$ otherwise) and $c_{jl} = 1$ if column j is in column cluster $\mathcal{C}_l$. According to this notation,

$$\mathbf{Q} = \mathbf{R}^\top \mathbf{P} \mathbf{C} \tag{3.3}$$

Equation 3.2 can be then rewritten as:

$$\hat{\tau}_{R|C}(\mathcal{R}, \mathcal{C}) = \sum_{k=1}^{K} \sum_{l=1}^{L} \frac{q_{kl}^2}{q_{\cdot l}} - \sum_{k=1}^{K} q_{k\cdot}^2$$

where $q_{k\cdot} = \sum_{l=1}^{L} \frac{t_{kl}}{A} = \sum_{l=1}^{L} \sum_{i \in \mathcal{R}_k} \sum_{j \in \mathcal{C}_l} \frac{a_{ij}}{A}$, and $q_{\cdot l} = \sum_{k=1}^{K} \frac{t_{kl}}{A} = \sum_{k=1}^{K} \sum_{i \in \mathcal{R}_k} \sum_{j \in \mathcal{C}_l} \frac{a_{ij}}{A}$. Moreover, since $q_{k\cdot} = \sum_{i \in \mathcal{R}_k} p_{i\cdot}$, and $p_{\cdot l} = \sum_{j \in \mathcal{C}_l} p_{\cdot j} = q_{\cdot l}$, the following formulation holds as well:

$$\hat{\tau}_{R|C}(\mathcal{R}, \mathcal{C}) = \sum_{k=1}^{K} \sum_{l=1}^{L} \left(\sum_{i \in \mathcal{R}_k} \frac{p_{il}}{p_{\cdot l}} \right) q_{kl} - \sum_{k=1}^{K} \left(\sum_{i \in \mathcal{R}_k} p_{i\cdot} \right) q_{k\cdot}$$

where $p_{i\cdot} = \sum_{j=1}^{m} \frac{a_{ij}}{A}$, $p_{\cdot j} = \sum_{i=1}^{n} \frac{a_{ij}}{A}$, $p_{il} = \sum_{j \in \mathcal{C}_l} p_{ij}$, and $p_{\cdot l} = \sum_{j \in \mathcal{C}_l} p_{\cdot j} = q_{\cdot l}$.

In [2], the authors propose an iterative strategy to find the best partition $\mathcal{R}$, when $\mathcal{C}$ is fixed (and conversely). Let $\mathbf{R}^{(t)}$ be the row cluster incidence matrix at iteration t, and $\mathbf{Q}^{(t)} = \mathbf{R}^{(t)\top} \mathbf{P} \mathbf{C}$ its associated distribution. The objective function $\hat{\tau}_{R|C}(\mathcal{R}^{(t)}, \mathcal{C})$ is

$$\hat{\tau}_{R|C}(\mathcal{R}^{(t)}, \mathcal{C}) = \sum_{i=1}^{n} \left(\sum_{l=1}^{L} \frac{p_{il}}{p_{\cdot l}} q_{kl}^{(t)} - p_{i\cdot} q_{k\cdot}^{(t)} \right) \tag{3.4}$$

$$\mathbf{A} = \left[\begin{array}{ccc|ccc} 2 & 3 & 1 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 2 & 2 & 3 \\ 0 & 0 & 1 & 0 & 5 & 2 \end{array}\right] \longrightarrow \mathbf{T} = \begin{bmatrix} 10 & 1 \\ 1 & 14 \end{bmatrix}$$

$$\mathbf{A}' = \left[\begin{array}{ccc|ccc} 2 & 3 & 1 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 1 & 2 & 2 & 3 \\ 0 & 0 & 1 & 0 & 5 & 2 \end{array}\right] \longrightarrow \mathbf{T}' = \begin{bmatrix} 9 & 2 \\ 0 & 16 \end{bmatrix}$$

Figure 1: Two matrices and their co-clustering results.

Each row $\mathbf{q}_k^{(t)}$ of $\mathbf{Q}^{(t)}$ can be interpreted as a prototype of the k-th cluster of $\mathcal{R}^{(t)}$, and the following similarity function between any row $\mathbf{p}_i$ of $\mathbf{P}$ and $\mathbf{q}_k^{(t)}$ is defined:

$$\sigma\left(\mathbf{p}_i, \mathbf{q}_k^{(t)}\right) = \sum_{l=1}^{L} \frac{p_{il}}{p_{\cdot l}} q_{kl}^{(t)} - p_{i\cdot} q_{k\cdot}^{(t)} \tag{3.5}$$

It measures the similarity between a "point" p_i and a cluster prototype $q_r^{(t)}$. The objective function becomes

$$\hat{\tau}_{R|C}(\mathcal{R}^{(t)}, \mathcal{C}) = \sum_{i=1}^{n} \sigma\left(\mathbf{p}_i, \mathbf{q}_{k^\star}^{(t)}\right) \tag{3.6}$$

where $k^\star = \arg\max_k \left(\sigma\left(\mathbf{p}_i, \mathbf{q}_k^{(t)}\right)\right)$ is the cluster assignment maximizing function σ.

3.3 Information leakage through co-clustering In its general setting, a co-clustering of a matrix $\mathbf{A}$ is a pair $(\mathcal{R}, \mathcal{C})$, where $\mathcal{R}$ is a partition of the rows and $\mathcal{C}$ a partition of the columns of $\mathbf{A}$. As an example, consider the two data matrices $\mathbf{A}$ and $\mathbf{A}'$ in Fig. 1 recording the purchase habits of a group of customers of an online shop (the rows represent the customers and the columns are the items sold). The only difference between $\mathbf{A}$ and $\mathbf{A}'$ is that $0 = a_{33} \neq a'_{33} = 1$, i.e., in $\mathbf{A}'$ the third customer buys a unit of the third product. When applied to $\mathbf{A}$, the co-clustering algorithm returns two row clusters (one including the first two rows and the second including the last two rows) and two column clusters (one with the first three columns and the other containing all the remaining columns). When $\mathbf{A}'$ is considered, the only minimal change in the data matrix has two consequences: (1) now the third column is more similar to the last columns, thus the co-clustering algorithm will return a different column clustering; (2) the contingency matrix associated to the co-clustering solution slightly changes, from $\mathbf{T}$ to $\mathbf{T}'$. Clearly, publishing the contingency matrix reveals whether the dataset is $\mathbf{A}$ or $\mathbf{A}'$. But the co-clustering outcome could also provide a further harm for the individuals' privacy: the column clustering. For instance, an attacker could infer that an atypical user, the only one buying the first two products of matrix $\mathbf{A}$ in significant quantities, is included in a dataset just by looking at the column clustering results and noticing that these two products have been clustered together.

4 Differentially Private Co-Clustering

In this section, we present DP-τCC, a differentially private co-clustering method based on the de-normalized Goodman-Kruskal's τ. We first describe how to compute the contingency table and the cluster assignment functions in a differentially private way. Then, we present the overall algorithm, which satisfies ε-differential privacy.

4.1 Differentially private computation of the core functions Before introducing the details of the method, we recall that, at each iteration, Fast-τCC considers the partition in one mode (for instance, the column partition $\mathcal{C}$, with L clusters, obtained at the previous iteration) fixed, projects the probability matrix $\mathbf{P}$ of $\mathbf{A}$ in the lower dimensional space given by the column clustering. Finally it computes a new row partition by assigning each row i to the cluster k maximizing $\sigma(\mathbf{p}_i, \mathbf{q}_k)$. In the following, we will describe how to make the entire procedure differentially-private.

First, we can formalize the entire iteration as a function that, given the contingency table $\mathbf{T}$ associated to a co-clustering solution and the column clustering $\mathbf{C}$, takes a data matrix, computes the best row-cluster assignment $\mathbf{R}$ and returns the contingency table associated to the co-clustering $(\mathcal{R}, \mathcal{C})$. In symbols, the contingency table computation function is

$$\begin{aligned} f_{(\mathbf{T},\mathbf{C})} : \mathbb{N}^{n\times m} &\longrightarrow \mathbb{N}^{k\times l} \\ \mathbf{A} &\longmapsto \mathbf{T}^{new} \end{aligned} \tag{4.7}$$

where the generic element of $\mathbf{T}^{new}$ is

$$t_{kl}^{new} = \sum_{i|k^\star(i)=k} \sum_{j|l^\star(j)=l} a_{ij} \tag{4.8}$$

and, recalling that $a_{\cdot l} = t_{\cdot l}$ and $A = T$,

$$\begin{aligned} k^\star(i) &= \arg\max_k \left(\sigma(\mathbf{p}_i, \mathbf{q}_k)\right) \\ &= \arg\max_k \left(\sum_{l=1}^{L} \left(\frac{a_{il}}{a_{\cdot l}} \frac{t_{kl}}{T}\right) - \frac{a_{i\cdot}}{A} \frac{t_{k\cdot}}{T}\right) \\ &= \arg\max_k \left(\sum_{l=1}^{L} a_{il} \left(\frac{t_{kl}}{t_{\cdot l}} - \frac{t_{k\cdot}}{T}\right)\right) \\ &\stackrel{\text{def}}{=} \arg\max_k \left(\sigma(\mathbf{a}_i, \mathbf{t}_k)\right). \end{aligned} \tag{4.9}$$

Function $l^\star(j)$, for the column-cluster assignment, is computed similarly. Since the output of $f_{(\mathbf{T},\mathbf{C})}$ is numeric, we can use the Laplace mechanism to add noise to each cell of its output $\mathbf{T}^{new}$, with scale of the Laplace noise proportional to the sensitivity of function $f_{(\mathbf{T},\mathbf{C})}$.

Let $\mathbf{A}$ and $\mathbf{A}'$ be two neighboring matrices, i.e., there is only one entry uv such that $a'_{uv} = a_{uv} + 1$, while all the other entries are unchanged. We consider function $f_{(\mathbf{T},\mathbf{C})}$ as the composition of two different functions: the first one, $\kappa_{(\mathbf{T},\mathbf{C})}$ (resp. $\lambda_{(\mathbf{T},\mathbf{R})}$), computes the clustering assignment $\mathbf{R}$ (resp. $\mathbf{C}$) for each row (resp. column) of the data matrix. The second one, $f_{(\mathbf{R},\mathbf{C})}$, considers the row and column partitions as fixed and computes the new contingency table:

$$f_{(\mathbf{T},\mathbf{C})} : \mathbf{A} \xrightarrow{\kappa_{(\mathbf{T},\mathbf{C})}} \mathbf{R} \xrightarrow{f_{(\mathbf{R},\mathbf{C})}} \mathbf{T}^{new}$$

The cluster assignment function $\kappa_{(\mathbf{T},\mathbf{C})}$ can be made private via the exponential mechanism (see Section 3.1). For each row i, instead of publishing the correct cluster assignment $\kappa(i)$, we can extract a cluster label k from the set of all the cluster labels $\{1, 2, \cdots, K\}$, with probability of extraction proportional to the utility function $\sigma(\mathbf{a}_i, \mathbf{t}_k)$. To compute the sensitivity of the utility function, consider two row vectors $\mathbf{a}_u$ and $\mathbf{a}'_u$ differing in only one entry $a'_{uv} = a_{uv} + 1$. Then,

$$\begin{aligned}\Delta(\sigma) &= |\sigma(\mathbf{a}_u, \mathbf{t}_k) - \sigma(\mathbf{a}'_u, \mathbf{t}_k)| = \\ &= \left|\sum_{l=1}^{L} a_{ul}\left(\frac{t_{kl}}{t_{\cdot l}} - \frac{t_{k\cdot}}{T}\right) - \sum_{l=1}^{L} a'_{ul}\left(\frac{t_{kl}}{t_{\cdot l}} - \frac{t_{k\cdot}}{T}\right)\right| = \\ &= \left|\sum_{l=1}^{L}(a_{ul} - a'_{ul})\left(\frac{t_{kl}}{t_{\cdot l}} - \frac{t_{k\cdot}}{T}\right)\right| = \left|\frac{t_{kl^\star(v)}}{t_{\cdot l^\star(v)}} - \frac{t_{k\cdot}}{T}\right|.\end{aligned}$$

The last equality holds because $a_{ul} = a'_{ul}$ when $l^\star(v) \neq l$ and $|a_{ul} - a'_{ul}| = 1$ when $l^\star(v) = l$. Therefore

$$\begin{aligned}\Delta(\sigma) = \max_{l=1,\ldots,L}\Bigg(&\max_{k=1,\ldots,K}\left(\frac{t_{kl}}{t_{\cdot l}} - \frac{t_{k\cdot}}{T}\right) - \\ &- \min_{k=1,\ldots,K}\left(\frac{t_{kl}}{t_{\cdot l}} - \frac{t_{k\cdot}}{T}\right)\Bigg).\end{aligned} \tag{4.10}$$

and the cluster assignment for each row i can be chosen among the set of all clusters with probability proportional to $\exp\left(\frac{\varepsilon\cdot\sigma(\mathbf{a}_i,\mathbf{t}_k)}{\Delta(\sigma)}\right)$ [12]. Once the row cluster assignment has been computed preserving differential privacy, both row and column clustering are public information and the sensitivity of the contingency table computation function is:

$$\begin{aligned}\Delta(f_{(\mathbf{T},\mathbf{C})}) &= ||f_{(\mathbf{R},\mathbf{C})}(\mathbf{A}) - f_{(\mathbf{R},\mathbf{C})}(\mathbf{A}')||_1 = \\ &= \sum_{k=1}^{K}\sum_{l=1}^{L}|t^{new}_{kl}(\mathbf{A}) - t^{new}_{kl}(\mathbf{A}')| = \\ &= |t^{new}_{\kappa(u)\lambda(v)}(\mathbf{A}) - t^{new}_{\kappa(u)\lambda(v)}(\mathbf{A}')| = 1\end{aligned} \tag{4.11}$$

Hence, the contingency table can be computed in a differentially private way injecting Laplace noise with scale parameter $\frac{1}{\varepsilon}$. Notice that, when we express the function $f_{(\mathbf{T},\mathbf{C})}$ as the composition of two functions and transform each of the two components into differentially private functions, the composition theorem assures that the overall function still preserves differential privacy and the overall privacy level is the sum of the privacy levels of the two subfunctions [14]. So, in order to transform $f_{(\mathbf{T},\mathbf{C})}$ in a ε-differentially private function, we devote 90% of the privacy budget ε to the computation of the cluster assignment and the remaining 10% to the computation of the contingency table. This choice is motivated by the fact that the the cluster assignment function is much more sensitive than the other function and very low levels of privacy budget risk to destroy the quality of the assignment.

Algorithm 1: DP-τCC($\mathbf{A}, \varepsilon, \mathcal{I}, L_0$)

Input: A $n \times m$ matrix $\mathbf{A}$, the privacy budget ε, the total number of iterations $\mathcal{I}$, the number of initial row and column clusters (K_0, L_0)

Result: $\mathbf{T}$, $\mathbf{C}$

```
// Initialization
ε' ← ε/(2I);
ε1 ← ε' · 0.1; ε2 ← ε' · 0.9;
M, R^(0), C^(0) = RandomMatrix(n, m, K0, L0);
t ← 1;
C^(t) ← DP-updatePart(A^⊤, C^(0)⊤ M^⊤, ε2);
// Optimization
while t < I do
    if t > 1 then
        C^(t) ←
          DP-updatePart(A^⊤ R^(t−1), T^⊤, ε2);
    end if
    T ← DP-computeT(A, R^(t−1), C^(t), ε1);
    R^(t) ← DP-updatePart(AC^(t), T, ε2);
    T ← DP-computeT(A, R^(t), C^(t), ε1);
    t ← t + 1;
end while
```

Algorithm 2: DP-computeT($\mathbf{A}, \mathbf{R}, \mathbf{C}, \varepsilon$)

Input: A $n \times m$ matrix $\mathbf{A}$, row clustering $\mathbf{R}$, column clustering $\mathbf{C}$, privacy budget ε
Result: $\mathbf{T}^{new}$

```
// Initialize local variables
K ← #columns of R; L ← #columns of C;
T^priv ← R^⊤ A C;
T^new ← Lap(T^priv, 1/ε_1);
for k = 1, ..., K do
    for l = 1, ..., L do
        if t_kl^new < 0 then
            t_kl^new = 0;
        end if
    end for
    if ||t_k^new||_1 = 0 then
        delete row k from T^new;
        K ← K − 1;
    end if
end for
```

Algorithm 3: DP-updatePart($\mathbf{A}, \mathbf{T}, \varepsilon$)

Input: A $n \times L$ matrix $\mathbf{A}$, a $K \times L$ clustering matrix $\mathbf{T}$, privacy budget ε
Result: $\mathbf{R}^{new}$

```
Compute Δ(σ) using Equation (4.10);
R^new ← 0_{n,K};
for i = 1, ..., n do
    u ← 0_K;
    for k = 1, ..., K do
        u_k = σ(a_i, t_k^new);
    end for
    Sample k* ∈ {1, ..., K} with probability
      ∝ e^{ε_2 u_k / Δ(σ)};
    r_{ik*}^new ← 1;
end for
```

4.2 The DP-τCC algorithm The overall differentially private co-clustering method (DP-τCC) is described in Algorithms 1, while Algorithm 2 provides the pseudocode of the computation of the differentially-private co-clustering matrix $\mathbf{T}$. The single partition update procedure, instead, is presented in Algorithm 3. It is worth mentioning how the *RandomMatrix* function is implemented. Since a classic (random) initialization strategy would consume too much privacy budget, we adopt a solution inspired by the one used in [29]. It consists in generating a boolean $n \times m$ matrix $\mathbf{M}$ with K_0 and L_0 random row and column clusters $\mathbf{R}^{(0)}$ and $\mathbf{C}^{(0)}$. If $K_0 \leq L_0$ (resp. $L_0 \leq K_0$), the largest square submatrix of $\mathbf{A}$ is initialized with K_0 (resp. L_0) equally-sized block co-clusters of 1's. The remaining part of the matrix is initialized with $L_0 - K_0$ (resp. $K_0 - L_0$) blocks of 1's. Finally, a random 1% of the entries are switched. When Algorithm 3 is called the first time, it computes the column partition $\mathbf{C}^{(1)}$ by considering the random co-clustering matrix obtained by calculating $\mathbf{C}^{(0)\top}\mathbf{M}^{\top}$. Successively, Algorithm 2 computes $\mathbf{T}^{(1)}$ using $\mathbf{C}^{(1)}$ and $\mathbf{R}^{(0)}$. The overall algorithm proceeds by alternating the optimization of $\mathbf{R}$ and $\mathbf{C}$ until the maximum number of iterations $\mathcal{I}$ is reached. It can be proved that DP-τCC preserves ε-differential privacy (it follows from the iterative application of the composition theorems [14]).

4.3 Time complexity The time complexity of Algorithm 2 is in $O(nmK_tL_t + K_tL_t)$, where K_t and L_t are the numbers of row and column clusters at iteration t. The complexity of Algorithm 3, instead, is in $O(nK_tL_t)$, when applied to update the row partition, and $O(mK_tL_t)$, when applied to update column clusters, with the exception of the first call (line 6 of Algorithm 1), where its complexity is in $O(nmL_0)$. During each iteration, except for the first one, Algorithm 1 executes both Algorithms 2 and 3 twice. In theory, K_t and L_t decrease at each iteration from the initial K_0 and L_0. In practice, however, if K_0 and L_0 are set close to the expected number of clusters, the values of K_t and L_t are stable, and very close to the initial (and maximal) K_0 and L_0 values, all along the execution of Algorithm 1. Hence, overall, Algorithm 1 has complexity in $O(nmL_0 + \mathcal{I}(nK_0L_0 + mK_0L_0 + 2(nmK_0L_0 + K_0L_0)))$. Consequently, if we consider only the dominating term, we can assume that the overall complexity of DP-τCC is in $O(2\mathcal{I}nmK_0L_0)$ with $K_0 \ll n$ and $L_0 \ll m$.

As a final remark, it must be noted that, as many other differentially-private iterative methods, the overall number of iterations must be kept low. As a consequence, if the algorithm is initialized with high K_0 and L_0, there will not be sufficient time for the number of co-clusters to decrease to a reasonable level. Consequently, as all other differentially-private clustering algorithms, K_0 and L_0 must be set equal or close to the expected/desired number of clusters.

5 Experiments

In this section, we report the results of the experiments we have conducted on eight challenging real-world datasets to assess the effectiveness of DP-τCC.

As we do not have any direct competitor performing differentially-private co-clustering, we focus on the closely related task of high-dimensional data clustering and consider the following approaches, already discussed in Section 2. As baseline we con-

sider a differentially-private k-means algorithm (DP-KMeans [29]), using the implementation available in the `diffprivlib` Python library[1]. As more closely related competitor, we use the high-dimensional data clustering approach (DPHD) presented in [1], adopting the MATLAB implementation made available by the authors of the paper[2]. Finally, we consider the Python implementation of the most recent approach for differentially-private NMF (DPNMF) described in [26][3]. It is worth noting that, as we said in Section 2, this last method has a different setting, as part of the computation, including the initialization (a crucial step for all NMF methods), is not performed in a privacy-preserving way. Although this choice puts DPNMF in the most favorable conditions of all tested algorithms, we will see that, in practice, it does not take any significant benefit from this. The maximum number of iterations is set to 4 for all the differentially-private algorithms (with the only exception of DP-KMeans, which determines the number of iterations from the size of the input data). We observed that, for a given ε, increasing the number of iterations does not bring any benefits to the accuracy of the methods. All methods are applied with increasing values of the privacy budget from $\varepsilon = 0.1$ to $\varepsilon = 3$. Since DPNMF satisfies (ε, δ)-differential privacy (with $\varepsilon < 1$), only for this algorithm, we set $\delta = 10^{-5}$ and let ε vary between 0.1 and 0.99999.

In all the competing approaches, when needed, we have modified the sensitivity to make it embrace our definition of neighboring datasets. Note that this choice always implies a diminished sensitivity and, consequently, a minor injection of noise. Moreover, since all the approaches (including ours) return differentially-private cluster prototypes, we perform an additional step for assigning the data objects (the rows of the matrix) to the closest prototype. Notice that, although this step violates differential privacy, it is used with the only purpose of performance evaluation on the ground truth. As non private reference, we use the authors' Python implementation of Fast-τCC [2][4]. Finally, as the number of clusters is required by all algorithms, it is set to the number of expected clusters on rows. Fast-τCC and DP-τCC also require the number of column clusters. Since there is no ground truth for columns, we set this number equal to the number of row clusters.

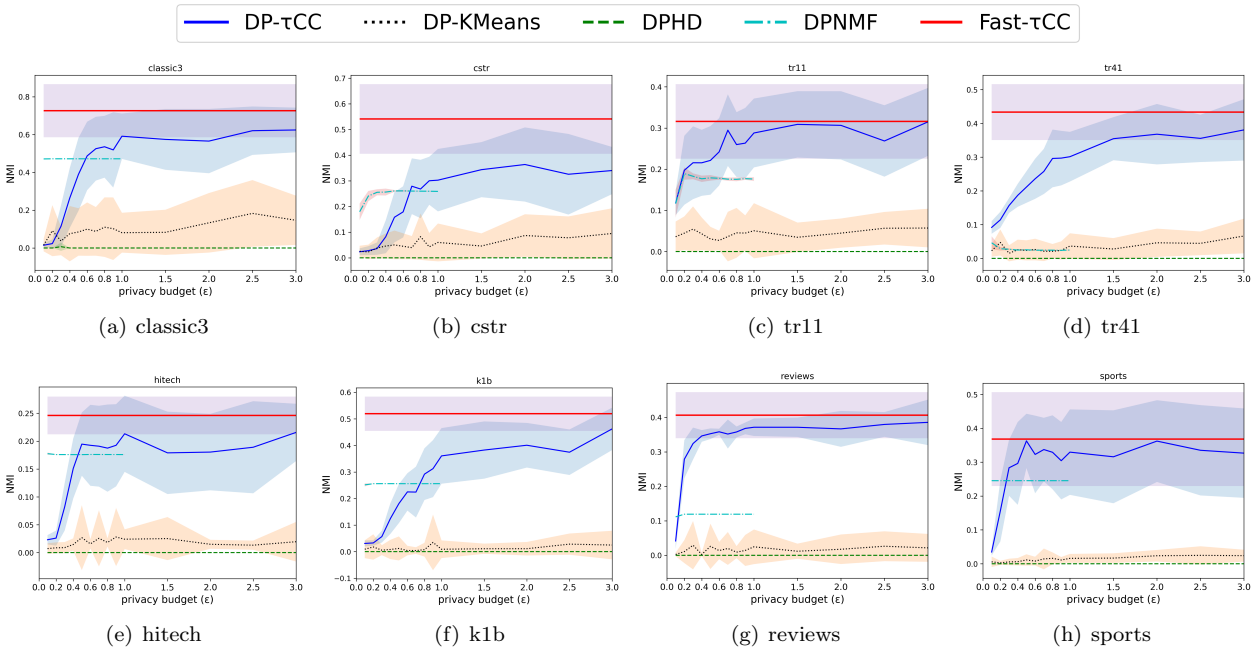
We assess the quality of the clustering solution through the average Normalized Mutual Information (NMI) computed on ten runs. All experiments are executed on a Linux server with 32 Intel Xeon Skylake cores running at 2.1GHz, 256GB RAM, and one Tesla T4 GPU. The source code and the dataset for reproducing all the experiments are available online[5].

Table 1: Dataset characteristics.

Dataset	# Rows	# Columns	% Density	# Classes
classic3	3 891	4 303	1.053	3
cstr	475	1 000	3.366	4
tr11	414	6 429	4.381	9
tr41	878	7,454	2.621	10
hitech	2 301	10 080	1.429	6
k1b	2 340	10 431	1.372	6
reviews	4 069	18 483	1.009	5
sports	8 280	14 870	0.856	7

5.1 Results on real-world matrices We apply all the competitors on high-dimensional real world datasets typically used in the validation of co-clustering algorithms. More in detail, we consider eight document-words co-occurrences matrices available in the CLUTO homepage[6]. Table 1 summarizes the main characteristics of these datasets, while Fig. 2 shows the average NMI of the results (computed on the row partition) for increasing values of ε and all the competing algorithms. In general, the plots show an ascending curve, typical of any differentially-private algorithm, but the behavior are very different among all the datasets. DP-Kmeans and DPHD, in fact, struggle in achieving performances that are comparable with those of DP-τCC and DP-NMF. With some exceptions, DPNMF outperforms DP-τCC only for small value of ε, but this is essentially due to the fact that its initialization, although randomized, is not privacy-preserving, as it actually accesses to the original data. In most cases, DP-τCC approaches the best performances when $0.5 \leq \varepsilon \leq 1$, an interval that provides reasonable privacy guarantees. As expected, the results get closer to the ones of Fast-τCC for larger datasets. It is well known, in fact, that most differentially-private algorithms give their best when applied to very large datasets. What is less obvious is that with only four iterations DP-τCC gets competitive performances w.r.t. its non private counterpart. Another general observation is that the performances seem capped to a maximum that never touches the optimal values of NMI. To explain this behavior, we recall that, differently from Fast-τCC, our algorithm performs a blind initialization (i.e., the initial co-clustering matrix is built without even looking at the data) and, most importantly, it is allowed to perform much less iterations. As a consequence, DP-τCC is more likely to get stuck in suboptimal solutions.

[1] `https://github.com/IBM/differential-privacy-library`
[2] `https://github.com/mouwenlong/dp-clustering-icml17`
[3] `https://github.com/swapnil-saha/NMF_DP`
[4] `https://github.com/rupensa/tauCC`
[5] `https://github.com/rupensa/dptaucc`
[6] `http://glaros.dtc.umn.edu/gkhome/views/cluto`



(a) classic3 (b) cstr (c) tr11 (d) tr41

(e) hitech (f) k1b (g) reviews (h) sports

Figure 2: NMI reported for all competitors on real-world datasets and for increasing values of ε.

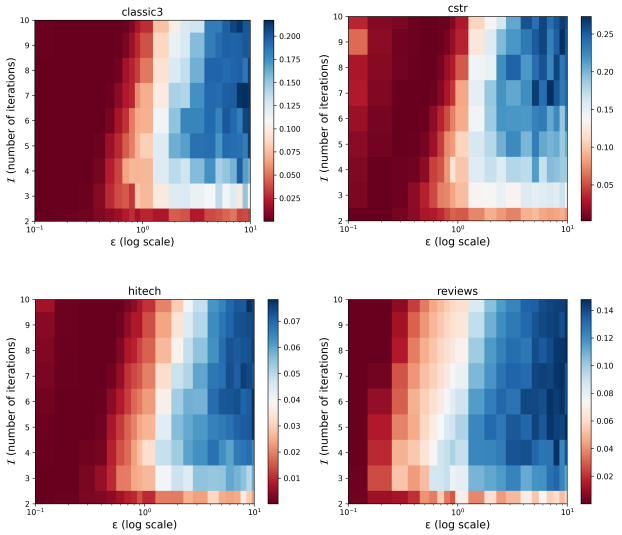


Figure 3: $\hat{\tau}_{R|C}$ w.r.t. the number of iterations (from 2 to 10) and the values of ε (from 0.1 to 10, in log scale).

5.2 Impact of the number of iterations To investigate the role of the number of iterations in the co-clustering quality, we measure the objective function value ($\hat{\tau}_{R|C}$) on the privacy-preserving contingency table released by DP-τCC by varying both ε and $\mathcal{I}$ (the maximum number of iterations). Fig. 3 reports the results obtained on 4 representative datasets: classic3, cstr, hitech and reviews. The behavior is always the same: clearly, high values of both ε and $\mathcal{I}$ results in higher co-clustering quality, in terms of $\hat{\tau}_{R|C}$ computed on the released $\mathbf{T}$. However, good trade-offs in terms of privacy and quality are obtained when $3 \leq \mathcal{I} \leq 5$. This observation holds for all other datasets as well.

6 Conclusion

We have introduced a privacy-preserving co-clustering method optimizing a simplified version of the Goodman-Kruskal's τ association measure. When compared to state-of-the-art high-dimensional and privacy preserving clustering approaches, our algorithm shows its effectiveness even for reasonably low values of the privacy budget ε. A limitation of our approach lies in the initialization, which is completely "blind", as it does not access the data to preserve the privacy budget. However, this step is crucial and we will investigate more accurate solutions as future work. Furthermore, we will try to increase the accuracy of the results by investigating ways to find the most suitable number of iterations based on ε, and to optimize the allocation of privacy budget to the different iterations and functions.

Acknowledgements The work presented in this paper is funded by the European Union – Next Generation EU, Mission 4 Component 2 Investment 1.1 CUP D53D23022370001 (GA n. P2022MSMAW).

References

[1] M.-F. Balcan, T. Dick, Y. Liang, W. Mou, and H. Zhang, *Differentially private clustering in high-dimensional euclidean spaces*, in Proc. of ICML 2017, 2017, pp. 322–331.

[2] E. Battaglia, F. Peiretti, and R. G. Pensa, *Fast parameterless prototype-based co-clustering*, Mach. Learn., 113 (2024), pp. 2153–2181.

[3] E. Battaglia, F. Peiretti, and R. G. Pensa, *Co-clustering: A survey of the main methods, recent trends, and open problems*, ACM Comput. Surv., 57 (2025), pp. 48:1–48:33.

[4] T. Benkhelif, F. Fessant, F. Clérot, and G. Raschia, *Co-clustering for differentially private synthetic data generation*, in Proc. of PAP@ECML PKDD 2017, 2017, pp. 36–47.

[5] A. Berlioz, A. Friedman, M. A. Kaafar, R. Boreli, and S. Berkovsky, *Applying differential privacy to matrix factorization*, in Proc. of ACM RecSys 2015, 2015, pp. 107–114.

[6] C. Biernacki, J. Jacques, and C. Keribin, *A survey on model-based co-clustering: High dimension and estimation challenges*, J. Classif., 40 (2023), pp. 332–381.

[7] A. Blanco-Justicia, D. Sánchez, J. Domingo-Ferrer, and K. Muralidhar, *A critical review on the use (and misuse) of differential privacy in machine learning*, ACM Comput. Surv., 55 (2023), pp. 160:1–160:16.

[8] A. Blum, C. Dwork, F. McSherry, and K. Nissim, *Practical privacy: the sulq framework*, in Proc. of ACM PODS 2005, 2005, pp. 128–138.

[9] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, *Differentially private empirical risk minimization.*, J. Mach. Learn. Res., 12 (2011).

[10] P. Deng, T. Li, H. Wang, S. Horng, Z. Yu, and X. Wang, *Tri-regularized nonnegative matrix tri-factorization for co-clustering*, Knowl. Based Syst., 226 (2021), p. 107101.

[11] I. S. Dhillon, S. Mallela, and D. S. Modha, *Information-theoretic co-clustering*, in Proc. of ACM SIGKDD 2003, 2003, pp. 89–98.

[12] J. Dong, D. Durfee, and R. Rogers, *Optimal differential privacy composition for exponential mechanisms*, in Proc. of ICML 2020, 2020, pp. 2597–2606.

[13] C. Dwork, F. McSherry, K. Nissim, and A. Smith, *Calibrating noise to sensitivity in private data analysis*, in Proc. of TCC 2006, 2006, pp. 265–284.

[14] C. Dwork, A. Roth, et al., *The algorithmic foundations of differential privacy.*, Found. Trends Theor. Comput. Sci., 9 (2014), pp. 211–407.

[15] Z. Fan and X. Xu, *Apdpk-means: A new differential privacy clustering algorithm based on arithmetic progression privacy budget allocation*, in Proc. of HPCC/SmartCity/DSS 2019, 2019, pp. 1737–1742.

[16] D. Feldman, C. Xiang, R. Zhu, and D. Rus, *Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks*, in Proc. of ACM/IEEE IPSN 2017, 2017, pp. 3–16.

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

[18] Z. Huang and J. Liu, *Optimal differentially private algorithms for k-means clustering*, in Proc. of ACM PODS 2018, 2018, pp. 395–408.

[19] S. Jha, L. Kruger, and P. McDaniel, *Privacy preserving clustering*, in Proceedings ESORICS 2005, 2005, pp. 397–417.

[20] P. Kairouz, S. Oh, and P. Viswanath, *The composition theorem for differential privacy*, IEEE Trans. Inf. Theory, 63 (2017), pp. 4037–4049.

[21] Z. Liu, Y.-X. Wang, and A. Smola, *Fast differentially private matrix factorization*, in Proc. of ACM RecSys 2015, 2015, pp. 171–178.

[22] F. McSherry and K. Talwar, *Mechanism design via differential privacy*, in Proc. of IEEE FOCS 2007), 2007, pp. 94–103.

[23] T. Ni, M. Qiao, Z. Chen, S. Zhang, and H. Zhong, *Utility-efficient differentially private k-means clustering based on cluster merging*, Neurocomputing, 424 (2021), pp. 205–214.

[24] K. Nissim, S. Raskhodnikova, and A. Smith, *Smooth sensitivity and sampling in private data analysis*, in Proc. of ACM STOC 2007, 2007, pp. 75–84.

[25] S. R. M. Oliveira and O. R. Zaïane, *Privacy preserving clustering by data transformation*, J. Inf. Data Manag., 1 (2010), pp. 37–52.

[26] S. Saha and H. Imtiaz, *Privacy-preserving non-negative matrix factorization with outliers*, ACM Trans. Knowl. Discov. Data, 18 (2024), pp. 64:1–64:26.

[27] A. Salah, M. Ailem, and M. Nadif, *Word co-occurrence regularized non-negative matrix tri-factorization for text data co-clustering*, in Proc. of AAAI/IAAI/EAAI 2018, 2018, pp. 3992–3999.

[28] U. Stemmer and H. Kaplan, *Differentially private k-means with constant multiplicative error*, Proc. of NIPS 2018, 31 (2018).

[29] D. Su, J. Cao, N. Li, E. Bertino, M. Lyu, and H. Jin, *Differentially private k-means clustering and a hybrid approach to private optimization*, ACM Trans. Priv. Secur., 20 (2017), pp. 1–33.

[30] J. Vaidya, Y. M. Zhu, and C. W. Clifton, *Privacy and data mining*, Springer, 2006.

[31] H. Wang, Y. Song, W. Chen, Z. Luo, C. Li, and T. Li, *A survey of co-clustering*, ACM Trans. Knowl. Discov. Data, 18 (2024), pp. 224:1–224:28.

[32] L. Wang, B. Zhao, and M. Kolar, *Differentially private matrix completion through low-rank matrix factorization*, in Proc. of AISTATS 2023, 2023.

[33] Y. Wang, Y.-X. Wang, and A. Singh, *Differentially private subspace clustering*, Proc. of NIPS 2005, 28 (2015).

[34] S. Zhang, L. Liu, Z. Chen, and H. Zhong, *Probabilistic matrix factorization with personalized differential privacy*, Knowl.-Based Syst., 183 (2019).