Biclustering with Alternating K-Means

Nicolas Fraiman
fraiman@email.unc.edu

Zichao Li
lizichao@live.unc.edu

Abstract

Biclustering is the task of simultaneously clustering the rows and columns of the data matrix into different subgroups such that the rows and columns within a subgroup exhibit similar patterns. In this paper, we consider the case of producing block-diagonal biclusters. We provide a new formulation of the biclustering problem based on the idea of minimizing the empirical clustering risk. We develop and prove a consistency result with respect to the empirical clustering risk. Since the optimization problem is combinatorial in nature, finding the global minimum is computationally intractable. In light of this fact, we propose a simple and novel algorithm that finds a local minimum by alternating the use of an adapted version of the k-means clustering algorithm between columns and rows. We evaluate and compare the performance of our algorithm to other related biclustering methods on both simulated data and real-world gene expression data sets. The results demonstrate that our algorithm is able to detect meaningful structures in the data and outperform other competing biclustering methods in various settings and situations.

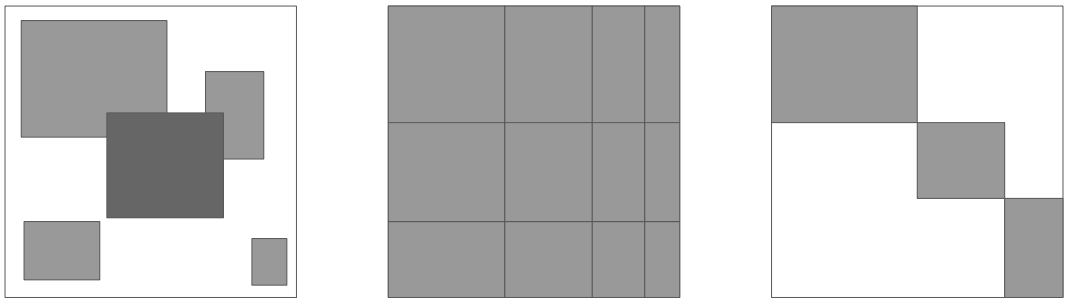
1 Introduction

In many fields of application, the data can be represented by a matrix, and people are interested in the task of simultaneously clustering the rows and columns of the data matrix into different subgroups such that the rows and columns within a subgroup exhibit similar patterns. This general task has been studied in many different application domains. For example, in gene expression analysis, people seek to identify subgroups of genes that have similar expression levels within corresponding subgroups of conditions [Cheng and Church, 2000]. In text mining, people attempt to recognize subgroups of documents that have similar properties with respect to corresponding subgroups of words [Dhillon, 2001]. In collaborative filtering, people wish to detect subgroups of customers with similar preferences toward corresponding subgroups of products [Hofmann and Puzicha, 1999]. The most common name of the task is biclustering [Cheng and Church, 2000, Tanay et al., 2002, Kluger et al., 2003, Prelić et al., 2006, Mankad and Michailidis, 2014], although it is also known by other names such as co-clustering [Dhillon, 2001, Dhillon et al., 2003, Cho et al., 2004, Banerjee et al., 2007, Chi et al., 2020], block clustering [Govaert and Nadif, 2003, 2005, 2008], and direct clustering [Hartigan, 1972].

Over the years, a large number of biclustering methods have been proposed. Comprehensive reviews of different biclustering methods can be found in Madeira and Oliveira [2004] and Tanay et al. [2005]. The biclustering methods could be classified into different groups based on the structure of the produced biclusters. Figure 1 shows three types of bicluster structures that could be obtained after appropriate row and column reordering:

1. In Figure 1a, the biclusters are arbitrarily positioned and can overlap with each other. The majority of the biclustering methods produce this type of biclusters, including Cheng and Church [2000], CTWC [Getz et al., 2000], ISA [Bergmann et al., 2003], SAMBA [Tanay et al., 2004], plaid models [Lazzeroni and Owen, 2002], OPSM [Ben-Dor et al., 2002], xMOTIFs [Murali and Kasif, 2003], and others [Li et al., 2009, Shabalin et al., 2009, Hochreiter et al., 2010].

2. In Figure 1b, the biclusters are non-overlapping and follow a checkerboard structure. The biclustering methods that produce this type of biclusters include spectral biclustering [Kluger et al., 2003], SSVD [Lee et al., 2010], sparse biclustering [Tan and Witten, 2014], convex biclustering [Chi et al., 2017],



(a) (b) (c)

Figure 1: Different types of bicluster structures (after row and column reordering). (a) Arbitrarily positioned overlapping biclusters, (b) non-overlapping biclusters with checkerboard structure, and (c) block-diagonal biclusters.

profile likelihood biclustering [Flynn and Perry, 2020], high-order spectral clustering [Han et al., 2020], and others [Dhillon et al., 2003, Govaert and Nadif, 2003, Cho et al., 2004, Chen et al., 2013].

3. In Figure 1c, the biclusters are rectangular diagonal blocks in the data matrix. In this case, there exist k mutually exclusive and exhaustive clusters of rows, and k corresponding mutually exclusive and exhaustive clusters of columns. Our method, named alternating k-means biclustering, produces this type of biclusters.

Some biclustering methods [Segal et al., 2001, Tang et al., 2001, Wang et al., 2002, Gu and Liu, 2008, Sill et al., 2011] produce other types of bicluster structures. A more detailed discussion is provided in Madeira and Oliveira [2004].

In general, methods that produce overlapping biclusters are more complex and flexible, whereas methods that produce non-overlapping biclusters are easier to interpret and visualize. Our method produces block-diagonal biclusters, which arise naturally in several applications. For example, in gene expression analysis, block-diagonal biclusters help people divide genes into different groups associated with different types of cancer tissues. In text mining, block-diagonal biclusters help people group documents into distinct clusters based on distinct sets of terms. One particular example is shown in Figure 2, where we apply our biclustering method to the breast and colon cancer gene expression data set [de Souto et al., 2008] consisting of 104 samples and 182 genes. As we can see, our biclustering method partitions samples into two groups that almost perfectly correspond to the two cancer types, and it also identifies two groups of genes associated with breast and colon cancer.

The key difference between clustering and biclustering is that clustering is based on *global* patterns whereas biclustering is based on *local* patterns. More specifically, when performing clustering on the rows of the data matrix, all the columns are taken into consideration. In contrast, when performing biclustering on the data matrix, the rows are clustered into different groups based on different subsets of columns. This characteristic of biclustering inspired us to develop a "local" version of k-means clustering: instead of performing k-means clustering on the rows using all the columns, we could consider only using different subsets of columns. In other words, the cluster centers could be defined using different subsets of columns instead of all the columns. Starting from this simple idea, we adapt the formulation and algorithm of k-means clustering to a biclustering method by making several important modifications, and the details of our formulation and algorithm are given in Section 2 and Section 5, respectively. Notably, our alternating k-means biclustering algorithm is conceptually as simple as k-means clustering, and it has the extra advantage of being able to discover local patterns as opposed to global patterns. These two characteristics make our algorithm an ideal candidate to serve as a baseline for biclustering problems even when our bicluster structure might not be flexible enough.

Our main contributions can be summarized as follows:

1. We provide a new formulation of the biclustering problem based on the idea of minimizing the empirical clustering risk. The formulation is adapted from the k-means clustering problem, with two important

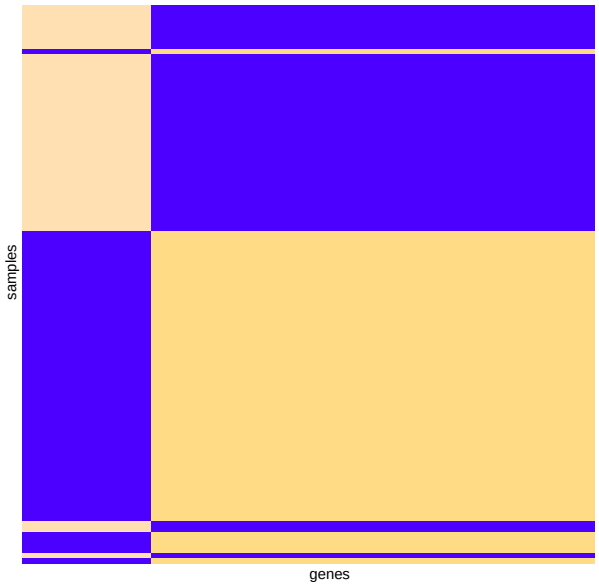


Figure 2: Heatmap of the estimated mean matrix from our biclustering method on the breast and colon cancer data set. The rows (samples) are ordered by true cancer type: first 42 are colon cancer tissues, and the other 62 are breast cancer tissues. The columns (genes) are reordered based on the produced biclusters for visualization purposes.

changes with respect to the definitions of cluster centers and norm. We further develop and prove a consistency result with respect to the empirical clustering risk, which is generally quite rare for biclustering methods.

2. Since minimizing the empirical clustering risk is a combinatorial optimization problem, finding the global minimum is computationally intractable. In light of this fact, we propose a simple and novel algorithm that finds a local minimum by alternately applying an adapted version of the k-means clustering algorithm between columns and rows. The simplicity of our algorithm makes it easy to understand, implement, and interpret. The R package `akmbiclust`, available on CRAN, implements our alternating k-means biclustering algorithm.

3. We empirically evaluate and compare the performance of our method to other related biclustering methods on both simulated data and real-world gene expression data sets. The empirical results have demonstrated that our method is able to detect meaningful structures in the data and outperform other competing biclustering methods in various settings and situations.

The rest of this paper is organized as follows. In Section 2, we formulate the task of biclustering as an optimization problem and present a consistency result. In Section 3, we provide a rigorous proof of the consistency result. In Section 4, we describe a probabilistic interpretation of the optimization problem. In Section 5, we present a simple algorithm that finds the local optimum by alternating the use of k-means clustering between columns and rows. In Section 6, we propose extending our method by adding penalization terms. In Section 7, we evaluate and compare the algorithm's performance on simulated data to other related biclustering algorithms. In Section 8, we apply the algorithm to three cancer gene expression data sets, demonstrating its advantage over other related biclustering algorithms in terms of sample misclassification rate. In Section 9, we conclude with a discussion.

2 Problem Formulation and Consistency Result

Suppose we have a $n \times m$ matrix $\mathbf{X}$ representing n data points $X_1, \ldots, X_n \in \mathbb{R}^m$. A typical example is the gene expression matrix, with rows corresponding to genes and columns corresponding to conditions. We formulate the task of biclustering on $\mathbf{X}$ as partitioning the n rows and m columns into k groups to get k biclusters, as shown in Figure 1c. More specifically, let $J = \{1, \ldots, n\}$ be the set of row indices, then J could be partitioned into k disjoint nonempty sets $J_1, \ldots, J_k$, where $J_1 \cup \cdots \cup J_k = J$. Similarly, let $I = \{1, \ldots, m\}$ be the set of column indices, then I could also be partitioned into k disjoint nonempty sets $I_1, \ldots, I_k$, where $I_1 \cup \cdots \cup I_k = I$. The k groups of row indices $J_1, \ldots, J_k$ and column indices $I_1, \ldots, I_k$ could be viewed as k biclusters. Note that under this definition of biclustering every row and column in the matrix $\mathbf{X}$ belongs to one and only one bicluster. In other words, the rows and columns in the biclusters are exhaustive and exclusive.

For simplicity of notation, we assume a vector is a row vector unless otherwise stated. For any $X = (x_1, \ldots, x_m) \in \mathbb{R}^m$, let $X(I_j) = (x_i)_{i \in I_j}$. For example, let $X = (1, 3, 4, 7)$, and $I_1 = \{1, 3\}, I_2 = \{2, 4\}$. Then $X(I_1) = (1, 4), X(I_2) = (3, 7)$. The space of $X(I_j)$ is defined as $\mathbb{R}^{I_j}$. We define a special norm on $\mathbb{R}^{I_j}$, called dimensionality-normalized norm. For any $X \in \mathbb{R}^{I_j}$, let $l_j = |I_j|$ denote the cardinality of the index set I_j, then it is also the dimension of the space $\mathbb{R}^{I_j}$, and the dimensionality-normalized norm of X is defined as

$$||X||_{dn} = \sqrt{\frac{\sum_{i \in I_j} x_i^2}{l_j}}.$$

The name "dimensionality-normalized norm" comes from the following simple relationship between the dimensionality-normalized norm and the Euclidean norm:

$$||X||_{dn}^2 = \frac{||X||_2^2}{l_j}.$$

Our method seeks to find the k groups of column indices $I_j, 1 \leq j \leq k$ and the k cluster centers $c_j \in \mathbb{R}^{I_j}, 1 \leq j \leq k$ such that the following objective function is minimized:

$$\sum_{i=1}^{n} \min_{1 \leq j \leq k} ||X_i(I_j) - c_j||_{dn}^2. \tag{1}$$

The corresponding k groups of row indices $J_t, 1 \leq t \leq k$ can be obtained by selecting all the rows that are "closest" to cluster center c_t:

$$J_t = \{i : \arg\min_{1 \leq j \leq k} ||X_i(I_j) - c_j||_{dn}^2 = t\}, 1 \leq t \leq k.$$

Note that here "closest" is measured by the distance function induced by the dimensionality-normalized norm:

$$\text{dist}(X_i(I_j), c_j) = ||X_i(I_j) - c_j||_{dn}, 1 \leq j \leq k, 1 \leq i \leq n.$$

Our biclustering method can be viewed as a more complicated version of the traditional k-means clustering, which only seeks to find the k cluster centers $c_j \in \mathbb{R}^m, 1 \leq j \leq k$ such that the following objective function is minimized:

$$\sum_{i=1}^{n} \min_{1 \leq j \leq k} ||X_i - c_j||_2^2.$$

However, it is important to note that there are two key differences:

1. The k cluster centers $c_1, \ldots, c_k$ in our objective function are not vectors in $\mathbb{R}^m$. Instead, $c_j \in \mathbb{R}^{I_j}$ for $1 \leq j \leq k$, and the k groups of column indices $I_1, \ldots, I_k$ also are parameters that we need to optimize over. In fact, finding the best column partition $I_1, \ldots, I_k$ is combinatorial in nature, which makes the optimization problem computationally intractable.

2. The norm in our objective function is not the Euclidean norm. Instead, it is the dimensionality-normalized norm.

The motivation of using the dimensionality-normalized norm instead of the Euclidean norm in the objective function (1) involves both theoretical and empirical aspects:

1. Theoretically, we provide a probabilistic interpretation of the objective function, in which dimensionality-normalized norm is the result of normalizing the log-likelihood to ensure fair comparison between vectors of different dimensionality. More details are given in Section 4.

2. Empirically, we observe that using Euclidean norm would often result in biclusters that are of undesirable shape. In the extreme case, when one uses Euclidean norm to produce two biclusters on a $n \times n$ matrix, the result would be one $1 \times (n-1)$ bicluster and one $(n-1) \times 1$ bicluster. This is because when using the Euclidean norm version of the objective function, extremely tall and thin bicluster combined with short and wide bicluster means many terms with a few columns and a few terms with many columns, and the result is a very small value of the objective function. In contrast, using dimensionality-normalized norm would produce biclusters of normal shape, and empirically it also performs much better than using Euclidean norm.

Suppose the data is a sequence of independent random observations $X_1, \ldots, X_n \in \mathbb{R}^m$ with the same distribution as a generic random variable X with distribution μ. We minimize the empirical clustering risk

$$W(\mathbf{I}, \mathbf{c}, \mu_n) = \frac{1}{n} \sum_{i=1}^{n} \min_{1 \le j \le k} ||X_i(I_j) - c_j||_{dn}^2 \tag{2}$$

over all possible choices of column partitions $\mathbf{I} = \{I_j\}_{1 \le j \le k}$ and cluster centers $\mathbf{c} = \{c_j\}_{1 \le j \le k}$. Here, μ_n is the empirical distribution of the data.

The performance of a clustering scheme given by the column partition $\mathbf{I}$ and cluster centers $\mathbf{c}$ is measured by the clustering risk

$$W(\mathbf{I}, \mathbf{c}, \mu) = \int \min_{1 \le j \le k} ||x(I_j) - c_j||_{dn}^2 d\mu(x). \tag{3}$$

The optimal clustering risk is defined as

$$W^*(\mu) = \inf_{\mathbf{I}} \inf_{\mathbf{c}} W(\mathbf{I}, \mathbf{c}, \mu). \tag{4}$$

Let $\delta_n \ge 0$. A column partition $\mathbf{I}_n$ and cluster centers $\mathbf{c}_n$ as a whole is a δ_n-minimizer of the empirical clustering risk if

$$W(\mathbf{I}_n, \mathbf{c}_n, \mu_n) \le W^*(\mu_n) + \delta_n,$$

where $W^*(\mu_n) = \inf_{\mathbf{I}} \inf_{\mathbf{c}} W(\mathbf{I}, \mathbf{c}, \mu_n)$. When $\delta_n = 0$, $\mathbf{I}_n$ and $\mathbf{c}_n$ as a whole is called an empirical risk minimizer. Since μ_n is supported on at most n points, the existence of an empirical risk minimizer is guaranteed.

The key theoretical result of this paper is the following consistency theorem, which states that the clustering risk of a δ_n-minimizer of the empirical clustering risk converges to the optimal risk as long as $\lim_{n\to\infty} \delta_n = 0$.

Theorem 1. *Assume that* $\mathbb{E}||X||_2^2 < \infty$. *Let* $\mathbf{I}_n$ *and* $\mathbf{c}_n$ *be a* δ_n*-minimizer of the empirical clustering risk. If* $\lim_{n\to\infty} \delta_n = 0$, *then*

1. $\lim_{n\to\infty} W(\mathbf{I}_n, \mathbf{c}_n, \mu) = W^*(\mu)$ *a.s., and*

2. $\lim_{n\to\infty} \mathbb{E}W(\mathbf{I}_n, \mathbf{c}_n, \mu) = W^*(\mu)$.

It is important to point out that we assume a minimizer of the empirical clustering risk can be found. However, finding the global minimum of the empirical clustering risk is a computationally intractable problem due to its combinatorial nature. In light of this fact, we present a simple algorithm in Section 5 that finds a local minimum based on the idea of alternating the use of k-means clustering between columns and rows.

3 Proof of the Consistency Result

Theorem 1 is quite similar to Proposition 2.1 in Biau et al. [2008], which showed that for k-means clustering, the clustering risk of a δ_n-minimizer of the empirical clustering risk converges to the optimal risk. More specifically, they used

$$W(\mathbf{c}, \mu_n) = \frac{1}{n}\sum_{i=1}^{n} \min_{1\le j\le k} ||X_i - c_j||_2^2,$$
$$W(\mathbf{c}, \mu) = \int \min_{1\le j\le k} ||x - c_j||_2^2 d\mu(x),$$
$$W^*(\mu) = \inf_{\mathbf{c}} W(\mathbf{c}, \mu).$$

in place of our $W(\mathbf{I}, \mathbf{c}, \mu_n)$, $W(\mathbf{I}, \mathbf{c}, \mu)$, and $W^*(\mu)$ defined in Equation (2), (3), and (4). Not surprisingly, the steps used to proved Proposition 2.1 in Biau et al. [2008] could also be adapted to prove Theorem 1, after making some minor changes and proving similar lemmas.

First, we introduce the definition of L_2 Wasserstein distance, which is essential for the proof. The L_2 Wasserstein distance between two probability measures μ_1 and μ_2 on $\mathbb{R}^m$, with finite second moment, is defined as

$$\gamma(\mu_1, \mu_2) = \inf_{X\sim\mu_1, Y\sim\mu_2} (\mathbb{E}||X - Y||_2^2)^{1/2},$$

where the infimum is taken over all joint distributions of two random variables X and Y such that X has distribution μ_1 and Y has distribution μ_2. Our main work is to prove the following lemma, which is similar to inequality (4) in Biau et al. [2008].

Lemma 1. *For any column partition* $\mathbf{I}$ *and cluster centers* $\mathbf{c}$*,*

$$\left|W(\mathbf{I}, \mathbf{c}, \mu_1)^{1/2} - W(\mathbf{I}, \mathbf{c}, \mu_2)^{1/2}\right| \le \gamma(\mu_1, \mu_2).$$

Proof. The proof is based on the proof of Lemma 3 in Linder [2002]. Let $X \sim \mu_1$ and $Y \sim \mu_2$ achieve the infimum defining $\gamma(\mu_1, \mu_2)$. Then

$$\begin{aligned}
W(\mathbf{I}, \mathbf{c}, \mu_1)^{1/2} &= \left[\int \min_{1\le j\le k} ||x(I_j) - c_j||_{dn}^2 d\mu_1(x)\right]^{1/2} \\
&= \left[\mathbb{E} \min_{1\le j\le k} ||X(I_j) - c_j||_{dn}^2\right]^{1/2} \\
&= \left[\mathbb{E} \min_{1\le j\le k} \frac{||X(I_j) - c_j||_2^2}{l_j}\right]^{1/2} \\
&\le \left[\mathbb{E} \min_{1\le j\le k} \frac{(||X(I_j) - Y(I_j)||_2 + ||Y(I_j) - c_j||_2)^2}{l_j}\right]^{1/2}.
\end{aligned}$$

Using Cauchy–Schwarz inequality, we have

$$\begin{aligned}
&\mathbb{E}\left[\frac{(||X(I_j) - Y(I_j)||_2 + ||Y(I_j) - c_j||_2)^2}{l_j}\right] \\
&= \mathbb{E}\left[\frac{||X(I_j) - Y(I_j)||_2^2}{l_j}\right] + \mathbb{E}\left[\frac{||Y(I_j) - c_j||_2^2}{l_j}\right] + 2\mathbb{E}\left[\frac{||X(I_j) - Y(I_j)||_2 ||Y(I_j) - c_j||_2}{l_j}\right] \\
&\le \mathbb{E}||X - Y||_2^2 + \mathbb{E}\left[\frac{||Y(I_j) - c_j||_2^2}{l_j}\right] + 2\mathbb{E}\left[||X - Y||_2 \cdot \frac{||Y(I_j) - c_j||_2}{\sqrt{l_j}}\right] \\
&\le \mathbb{E}||X - Y||_2^2 + \mathbb{E}\left[\frac{||Y(I_j) - c_j||_2^2}{l_j}\right] + 2\left[\mathbb{E}||X - Y||_2^2\right]^{1/2}\left[\mathbb{E}\frac{||Y(I_j) - c_j||_2^2}{l_j}\right]^{1/2} \\
&= \left(\left[\mathbb{E}||X - Y||_2^2\right]^{1/2} + \left[\mathbb{E}\frac{||Y(I_j) - c_j||_2^2}{l_j}\right]^{1/2}\right)^2.
\end{aligned}$$

Consequently

$$
\begin{aligned}
W(\mathbf{I},\mathbf{c},\mu_1)^{1/2} &\le \left[\mathbb{E}\min_{1\le j\le k}\frac{(||X(I_j)-Y(I_j)||_2+||Y(I_j)-c_j||_2)^2}{l_j}\right]^{1/2}\\
&\le \left[\mathbb{E}||X-Y||_2^2\right]^{1/2}+\left[\mathbb{E}\min_{1\le j\le k}\frac{||Y(I_j)-c_j||_2^2}{l_j}\right]^{1/2}\\
&= \left[\mathbb{E}||X-Y||_2^2\right]^{1/2}+\left[\mathbb{E}\min_{1\le j\le k}||Y(I_j)-c_j||_{dn}^2\right]^{1/2}\\
&= \gamma(\mu_1,\mu_2)+W(\mathbf{I},\mathbf{c},\mu_2)^{1/2},
\end{aligned}
$$

which implies that $W(\mathbf{I},\mathbf{c},\mu_1)^{1/2}-W(\mathbf{I},\mathbf{c},\mu_2)^{1/2}\le\gamma(\mu_1,\mu_2)$. The other direction can be proved similarly. □

Having proved Lemma 1, we are ready to prove the following lemma, which is similar to Lemma 4.1 in Biau et al. [2008].

Lemma 2. *Let $\mathbf{I}_n$ and $\mathbf{c}_n$ be a δ_n-minimizer of the empirical clustering risk. Then*

$$
W(\mathbf{I}_n,\mathbf{c}_n,\mu)^{1/2}-[\inf_{\mathbf{I}}\inf_{\mathbf{c}}W(\mathbf{I},\mathbf{c},\mu)]^{1/2}\le 2\gamma(\mu,\mu_n)+\sqrt{\delta_n}.
$$

Proof. After replacing $W(\mathbf{c}_n,\mu)$, $\inf_{\mathbf{c}}W(\mathbf{c},\mu)$ and some other intermediate variables with $W(\mathbf{I}_n,\mathbf{c}_n,\mu)$, $\inf_{\mathbf{I}}\inf_{\mathbf{c}}W(\mathbf{I},\mathbf{c},\mu)$ and other corresponding intermediate variables, the same steps that are used to prove Lemma 4.1 in Biau et al. [2008] could be applied to prove Lemma 2. Lemma 1 is used in the last inequality of the proof. □

Theorem 1 follows naturally from Lemma 2 and Lemma 4.2 in Biau et al. [2008], which states that $\lim_{n\to\infty}\gamma(\mu,\mu_n)=0$ a.s., and $\lim_{n\to\infty}\mathbb{E}\gamma^2(\mu,\mu_n)=0$.

4 A Probabilistic Interpretation of the Optimization Problem

In this section, we provide a probabilistic interpretation of the optimization problem, which also serves as the motivation behind the definition of the dimensionality-normalized norm. Suppose every row X_i in the matrix $\mathbf{X}$ is generated independently through the following process:

1. Select a nonempty subset of all the columns $I_j\subset I$, and the entries in those columns follow a multivariate normal distribution with mean vector c_j and covariance matrix $\sigma^2\mathbf{I}$ (here $\mathbf{I}$ denotes the identity matrix):
$$X_i(I_j)\sim\mathcal{N}(c_j,\sigma^2\mathbf{I}).$$
2. The entries in the other columns are considered as noise and do not affect the likelihood of X_i.

The log-likelihood of X_i has the following property:

$$
\log\mathcal{L}(I_j,c_j|X_i)\propto-\frac{1}{2\sigma^2}||X_i(I_j)-c_j||_2^2-\frac{l_j}{2}\log(2\pi\sigma^2),
$$

where $l_j=|I_j|$ is the cardinality of the index set I_j and also the dimensionality of the vector $X_i(I_j)$.

Naturally, the next step is to maximize the log-likelihood of X_i over I_j and c_j. However, there is one issue: different I_j might have different cardinality l_j. This means that $X_i(I_j)$ might have different dimensionality, and directly comparing the log-likelihood of vectors of different dimensionality is problematic: when $\log(2\pi\sigma^2)>0$, increasing the dimensionality l_j would monotonically decrease the log-likelihood.

One simple solution is to maximize the dimensionality-normalized log-likelihood of X_i:

$$
\frac{\log\mathcal{L}(I_j,c_j|X_i)}{l_j}\propto-\frac{||X_i(I_j)-c_j||_2^2}{l_j},
$$

which is equivalent to minimizing

$$\frac{||X_i(I_j) - c_j||_2^2}{l_j} = ||X_i(I_j) - c_j||_{dn}^2.$$

Therefore, maximizing the joint dimensionality-normalized log-likelihood of all the rows in the matrix $\mathbf{X}$ is equivalent to minimizing the empirical clustering risk

$$\frac{1}{n}\sum_{i=1}^{n} \min_{1\le j\le k} ||X_i(I_j) - c_j||_{dn}^2.$$

This equivalence establishes a connection between the optimization perspective and the probabilistic perspective of the biclustering problem.

5 Algorithm

In this section, we present a simple and novel algorithm that finds a local minimum of the empirical clustering risk. An implementation of our algorithm is provided in the R package `akmbiclust`, available on CRAN.

The idea is to alternate the use of an adapted version of the k-means clustering algorithm between columns and rows. Similar to the widely used Lloyd's algorithm in k-means clustering, our algorithm is also based on heuristics and does not guarantee to achieve global optimum. In each individual run, our alternating k-means biclustering algorithm works as described in Algorithm 1 below.

Algorithm 1 Alternating k-means biclustering

1. Start by performing k-means clustering separately on the rows and columns of the input matrix $\mathbf{X}$ to obtain the initial partitions of the n rows $J_1, \ldots, J_k$ and m columns $I_1, \ldots, I_k$. Calculate and record the loss.

2. With a fixed $I_1, \ldots, I_k$, the optimal cluster centers $c_1, \ldots, c_k$ could be found in the following way:

   (a) (Update step) Given row partitions $J_1, \ldots, J_k$, update the cluster centers $c_1, \ldots, c_k$ by the following equation:
   $$c_j = \frac{1}{|J_j|}\sum_{i\in J_j} X_i(I_j), 1 \le j \le k.$$

   (b) (Assignment step) Given cluster centers $c_1, \ldots, c_k$, update the row partitions $J_1, \ldots, J_k$ by assigning every row to the cluster center with the smallest distance (induced by the dimensionality-normalized norm), and all the rows that are closest to c_j form $J_j, 1 \le j \le k$.

   Alternate between (a) and (b) until convergence, and obtain a partition of the n rows $J_1, \ldots, J_k$.

3. Transpose the matrix $\mathbf{X}$, and $J_1, \ldots, J_k$ becomes a partition of the n columns. Again, alternate between (a) and (b) until convergence, and obtain a partition the m rows $I_1, \ldots, I_k$. Transpose the matrix $\mathbf{X}$ back, and $I_1, \ldots, I_k$ becomes a partition of the m columns.

4. Alternate between step 2 and step 3 until convergence. Calculate and record the loss.

5. Compare the losses at the end of step 1 and step 4. Output the k groups of row indices $J_1, \ldots, J_k$ and column indices $I_1, \ldots, I_k$ associated with the minimum loss.

Noticeably, the subroutine that alternates between (a) and (b) is quite similar to the widely used Lloyd's algorithm in k-means clustering, which is the reason why our algorithm is called "alternating k-means biclustering". However, we note that there are two important differences:

1. The cluster centers $c_1, \ldots, c_k$ are not vectors in $\mathbb{R}^m$. Instead, $c_j \in \mathbb{R}^{I_j}$ for $1 \le j \le k$.

2. When calculating the distance between a row X_i and a cluster center c_j, the distance function is not induced by the Euclidean norm. Instead, it is induced by the dimensionality-normalized norm.

It is recommended to run our algorithm multiple times and choose the result with the minimum loss, each time starting with a different initialization by randomly permuting the rows and columns of the input matrix $\mathbf{X}$. The reason is twofold:

1. First, our algorithm does not guarantee global optimum and depends on the partitions obtained from the initial separate k-means clustering, so running our algorithm multiple times increases the probability of finding a smaller local minimum.

2. Second, just like k-means clustering algorithm, our algorithm also might encounter empty cluster problem. More specifically, if in any assignment step any group of row indices $J_j, 1 \leq j \leq k$ becomes empty, then the algorithm cannot proceed and need to restart. The probability of having empty clusters is small when k is much smaller than $\min(n, m)$, but might become larger when k approaches $\min(n, m)$.

6 Penalization

In this section, we consider extending our method by adding penalization terms to the loss function. So far, the loss function that we minimize is the empirical clustering risk:

$$\frac{1}{n}\sum_{i=1}^{n}\min_{1\leq j\leq k}||X_i(I_j)-c_j||_{dn}^2.$$

Intuitively, minimizing this loss encourages all the rows in the bicluster j to have similar entries in the columns I_j across different rows. However, it does not differentiate between how large or small those entries are. Therefore, it might be beneficial to place some form of penalization on those entries to encourage "good" biclustering results. To this end, we consider the following penalization method:

Let $||\mathbf{X}||_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}X_{ij}^2}$ denote the Frobenius norm of a matrix $\mathbf{X}\in\mathbb{R}^{n\times m}$. Let the index of the k biclusters be from 0 to $k-1$. For every bicluster j where $0\leq j\leq k-1$, let $\mathbf{X}^{(j)}$ denote the submatrix of $\mathbf{X}$ consisting only of rows and columns that belong to bicluster j. The following penalization term is added to every bicluster j where $1\leq j\leq k-1$:

$$\lambda\cdot\frac{||\mathbf{X}||_F^2}{||\mathbf{X}^{(j)}||_F^2+1}.$$

Note that bicluster 0 is the special bicluster that does not have the above penalization term. The parameter λ is a tuning parameter.

The motivation of the penalization method comes from the following two observations:

1. In general, the entries in the biclusters should represent signals, which means that they should not be close to zero. Therefore, the Frobenius norm of the submatrix $||\mathbf{X}^{(j)}||_F$ induced by the bicluster j should be large.

2. However, not all rows and columns should be classified into one of k biclusters representing signals. Some rows or columns might just consist of random noise, and it is reasonable to include a special bicluster that represents random noise. Adding the penalization term $\lambda\cdot\frac{||\mathbf{X}||_F^2}{||\mathbf{X}^{(j)}||_F^2+1}$ to every bicluster creates a potential problem: when there is supposed to be a bicluster with entries close to zero, the penalization term might become excessively large. Thus, we choose to not apply the penalization term to bicluster 0, which is the special bicluster representing noise.

In the end, we decide to use the above penalization function, because it works reasonably well on both simulated and real-world data.

The penalized loss function can be written as:

$$\frac{1}{n}\sum_{i=1}^{n}\min_{0\leq j\leq k-1}||X_i(I_j)-c_j||_{dn}^2+\lambda\sum_{j=1}^{k-1}\frac{||\mathbf{X}||_F^2}{||\mathbf{X}^{(j)}||_F^2+1}.$$

When $\lambda = 0$, the penalized loss function reduces to the empirical clustering risk.

It is important to point out that the loss function does not affect the alternating process between step 2 and step 3 in Algorithm 1. However, the loss function does affect which specific biclustering result is chosen among different biclustering results produced by the algorithm: typically the algorithm is run with many random initializations, and even in each individual run, the algorithm needs to compare the two losses of two biclustering results: one at the end of step 1 corresponding to the partitions obtained by separate k-means, one at the end of step 4 when the algorithm finishes performing alternating k-means. Among all the different biclustering results, the biclustering result with the minimum loss is chosen as the final output.

7 Simulation Studies

In this section, we evaluate and compare the performance of the following five biclustering methods on simulated data with different settings:

1. Alternating k-means biclustering (AKM): This is the method presented in this paper. We use the penalized loss function in Section 6, with three different λ values: 0, 0.1, and 1.

2. Separate k-means clustering (KM): This method simply performs k-means clustering separately on the rows and columns.

3. Profile likelihood biclustering (PL) [Flynn and Perry, 2020]: This method is based on profile likelihood and has associated consistency guarantees. We implement the method using the R package `biclustpl`. The distribution family is selected as Gaussian, which is the true distribution of the simulated data.

4. Sparse biclustering (SBC) [Tan and Witten, 2014]: This method assumes the entries are normally distributed with a bicluster-specific mean and a common variance, and maximizes the L_1-penalized log-likelihood to obtain sparse biclusters. We implement the method using the R package `sparseBC`. The input matrix is always mean-centered before applying the method, and the tuning parameter λ is selected by choosing the λ with the smallest BIC over a grid of λ values, both of which are suggested in their paper.

5. High-order spectral clustering (HSC) [Han et al., 2020]: This method assumes a tensor block model and provides statistical optimality guarantees under a mild sub-Gaussian noise assumption. We implement the method using the R package `HLloyd`.

All methods except HSC are run with 100 random initializations.

Among numerous existing biclustering methods, the above five methods are selected to evaluate and compare their performance in the simulation studies because they satisfy the following two requirements:

1. Every row should be classified into one and only one row cluster. In addition, every column should also be classified into one and only one column cluster. This means that the biclustering methods should produce non-overlapping biclusters with checkerboard structure (Figure 1b) or block-diagonal biclusters (Figure 1c).

2. In addition, the biclustering methods should also allow explicitly specifying the number of clusters that the rows and columns are classified into. A few biclustering methods such as spectral biclustering [Kluger et al., 2003], SSVD [Lee et al., 2010], and convex biclustering [Chi et al., 2017] satisfy the first requirement but do not satisfy this requirement.

In our simulations, the evaluation metric is the misclassification rate, which is defined as:

$$\text{misclassification rate} = \frac{\text{number of entries classified into the wrong row or column cluster}}{\text{total number of entries in the input matrix } \mathbf{X}}.$$

Smaller misclassification rate indicates better performance, and a perfect biclustering result would have a misclassification rate of 0.

We generate simulated data in three different settings. In all three settings, the input matrix $\mathbf{X}$ is generated using a 2×2 block model, and for all the methods we set the number of clusters that the rows and

columns are classified into to be 2. The number of rows is set to be $n = 400$, and the number of columns is set to be $m = a \cdot n$ where $a \in \{0.5, 1.0, 2.0\}$. The entries X_{ij} in the input matrix $\mathbf{X}$ are generated independently through the following process:

1. Sample the true row class $u_i \in \{1, 2\}$ from the multinomial distribution with probability $p = (0.3, 0.7)$.
2. Sample the true column class $v_j \in \{1, 2\}$ from the multinomial distribution with probability $q = (0.2, 0.8)$.
3. Conditioning on u_i and v_j, X_{ij} follows a Gaussian distribution with mean $\mathbf{M}_{u_i v_j}$ and standard deviation $\mathbf{S}_{u_i v_j}$:
$$X_{ij}|u_i, v_j \sim \mathcal{N}(\mathbf{M}_{u_i v_j}, \mathbf{S}^2_{u_i v_j}).$$

Note that $\mathbf{M}$ and $\mathbf{S}$ are 2×2 matrices representing the means and standard deviations of entries in different blocks. They are different for each simulation setting.

7.1 Simulation 1: Blocks with Different Means and the Same Variance

In the first simulation, we consider the case where the 2×2 blocks have different means and the same variance. More specifically, we set
$$\mathbf{M} = b \cdot \begin{bmatrix} 0.36 & 0.90 \\ -0.58 & -0.06 \end{bmatrix},$$
where $b \in \{0.20, 0.25, 0.30\}$. The entries of the matrix are simulated from a uniform distribution on $[-1, 1]$. As b increases, the difference between the means in different blocks also increases. In addition, we set
$$\mathbf{S} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$
which means that all entries have the same standard deviation of 1. This type of structure is exactly what many biclustering methods including PL, SBC, and HSC assume the input matrix $\mathbf{X}$ has, therefore we would expect their performance to be good.

	AKM ($\lambda = 0$)	AKM ($\lambda = 0.1$)	AKM ($\lambda = 1$)	KM	PL	SBC	HSC
				$b = 0.20$			
$a = 0.5$	0.722(0.002)	0.591(0.015)	0.527(0.007)	0.542(0.006)	0.328(0.010)	0.409(0.014)	0.387(0.010)
$a = 1.0$	0.723(0.002)	0.470(0.006)	0.474(0.006)	0.489(0.006)	0.244(0.005)	0.258(0.008)	0.313(0.006)
$a = 2.0$	0.718(0.002)	0.445(0.006)	0.447(0.006)	0.444(0.006)	0.221(0.004)	0.222(0.004)	0.299(0.004)
				$b = 0.25$			
$a = 0.5$	0.715(0.003)	0.457(0.010)	0.450(0.007)	0.467(0.007)	0.201(0.007)	0.223(0.012)	0.251(0.007)
$a = 1.0$	0.716(0.003)	0.415(0.010)	0.415(0.010)	0.404(0.007)	0.150(0.006)	0.151(0.006)	0.222(0.006)
$a = 2.0$	0.706(0.002)	0.347(0.013)	0.338(0.013)	0.321(0.006)	0.147(0.004)	0.148(0.004)	0.223(0.005)
				$b = 0.30$			
$a = 0.5$	0.711(0.003)	0.392(0.012)	0.387(0.011)	0.386(0.010)	0.110(0.004)	0.110(0.004)	0.166(0.006)
$a = 1.0$	0.694(0.003)	0.291(0.015)	0.270(0.013)	0.287(0.010)	0.083(0.004)	0.079(0.004)	0.140(0.006)
$a = 2.0$	0.670(0.004)	0.173(0.007)	0.169(0.005)	0.197(0.007)	0.080(0.003)	0.081(0.003)	0.143(0.006)

Table 1: The means (and standard errors) of the misclassification rate for Simulation 1 over 50 simulations.

Results are reported in Table 1. Under this setting, we see that PL and SBC have similar misclassification rates, followed by HSC, and all three are much smaller than the other four methods. KM, AKM with $\lambda = 1$ and $\lambda = 0.1$ also have similar misclassification rates, though they are significantly larger than PL, SBC, and HSC. AKM with $\lambda = 0$ has the worst performance, with misclassification rates around 0.7 in all cases. In addition, we observe a general trend that as a and b increase, the misclassification rates decrease. This trend agrees with our expectation, because larger a means larger input matrix, and larger b means larger difference between the means in different blocks, both of which should improve the performance of biclustering methods.

7.2 Simulation 2: Blocks with Different Variances and the Same Mean

In the second simulation, we consider the case where the 2×2 blocks have different variances and the same mean. More specifically, we set

$$\mathbf{S} = \begin{bmatrix} 1 & 1+b \\ 1+b & 1 \end{bmatrix},$$

where $b \in \{0.20, 0.25, 0.30\}$. As b increases, the difference between the standard deviations in different blocks also increases. In addition, we set

$$\mathbf{M} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

which means that all entries have the same mean of 0. This is the case where the blocks are defined not by different means but by different variances, and many biclustering methods including PL, SBC, and HSC are unable to detect this type of structure.

	AKM ($\lambda = 0$)	AKM ($\lambda = 0.1$)	AKM ($\lambda = 1$)	KM	PL	SBC	HSC
				$b = 0.20$			
$a = 0.5$	0.474(0.015)	0.422(0.018)	0.484(0.027)	0.717(0.002)	0.717(0.002)	0.701(0.010)	0.722(0.002)
$a = 1.0$	0.123(0.019)	0.055(0.004)	0.053(0.003)	0.728(0.002)	0.726(0.002)	0.716(0.008)	0.729(0.002)
$a = 2.0$	0.015(0.001)	0.015(0.001)	0.016(0.001)	0.726(0.002)	0.724(0.002)	0.729(0.002)	0.729(0.002)
				$b = 0.25$			
$a = 0.5$	0.079(0.009)	0.067(0.007)	0.070(0.006)	0.717(0.002)	0.722(0.002)	0.708(0.008)	0.722(0.002)
$a = 1.0$	0.014(0.001)	0.013(0.001)	0.015(0.001)	0.726(0.002)	0.722(0.002)	0.715(0.008)	0.728(0.002)
$a = 2.0$	0.003(0.000)	0.003(0.000)	0.004(0.000)	0.722(0.002)	0.720(0.002)	0.719(0.007)	0.727(0.002)
				$b = 0.30$			
$a = 0.5$	0.025(0.002)	0.025(0.001)	0.045(0.011)	0.718(0.002)	0.716(0.002)	0.716(0.006)	0.722(0.002)
$a = 1.0$	0.003(0.000)	0.003(0.000)	0.005(0.001)	0.724(0.003)	0.725(0.002)	0.711(0.009)	0.725(0.002)
$a = 2.0$	0.000(0.000)	0.000(0.000)	0.001(0.000)	0.718(0.002)	0.716(0.002)	0.716(0.008)	0.723(0.002)

Table 2: The means (and standard errors) of the misclassification rate for Simulation 2 over 50 simulations.

Results are reported in Table 2. Under this setting, we see that KM, PL, SBC, and HSC all have similarly bad performance, with misclassification rates around 0.72 in all cases. In contrast, AKM with $\lambda = 0$, $\lambda = 0.1$, and $\lambda = 1$ have similar and much better performance, and in some cases ($a \in \{1.0, 2.0\}, b \in \{0.25, 0.30\}$) even produce near perfect biclustering results. In addition, with regard to AKM, we also see the general trend that as a and b increase, the misclassification rates decrease. Interestingly, in this setting larger b means larger difference between the variances in different blocks, and only AKM is able to detect and respond to this type of structure.

7.3 Simulation 3: Blocks with Different Means and Different Variances

In the third simulation, we consider the case where the 2×2 blocks have different means and different variances, which is a combination of the first and second case. More specifically, we set

$$\mathbf{M} = b \cdot \begin{bmatrix} 0.36 & 0.90 \\ -0.58 & -0.06 \end{bmatrix}, \ \mathbf{S} = \begin{bmatrix} 1 & 1+b \\ 1+b & 1 \end{bmatrix},$$

where $b \in \{0.20, 0.25, 0.30\}$. As b increases, the difference between the means and standard deviations in different blocks also increases. This type of structure is arguably the most common type in practice, where different biclusters not only have different means but also have different variances.

Results are reported in Table 3. Under this setting, we again observe that AKM with $\lambda = 0$, $\lambda = 0.1$, and $\lambda = 1$ have very similar performance, and clearly they have much smaller misclassification rates compared to other methods in all cases.

Comparing the results in Table 3 to those in Table 1, we see that the additional difference between the variances in different blocks significantly benefits AKM with $\lambda = 0$, $\lambda = 0.1$, and $\lambda = 1$, resulting in a drastic decrease in misclassification rates. In contrast, this additional difference between block variances harms the performance of KM, PL, SBC, and HSC, causing varying degrees of increase in misclassification rates. This

	AKM ($\lambda = 0$)	AKM ($\lambda = 0.1$)	AKM ($\lambda = 1$)	KM	PL	SBC	HSC
				$b = 0.20$			
$a = 0.5$	0.282(0.018)	0.236(0.016)	0.244(0.018)	0.561(0.006)	0.389(0.012)	0.444(0.012)	0.439(0.009)
$a = 1.0$	0.036(0.002)	0.034(0.002)	0.035(0.002)	0.502(0.005)	0.287(0.006)	0.285(0.008)	0.366(0.005)
$a = 2.0$	0.012(0.001)	0.012(0.001)	0.012(0.001)	0.463(0.005)	0.263(0.005)	0.259(0.005)	0.349(0.004)
				$b = 0.25$			
$a = 0.5$	0.045(0.004)	0.042(0.003)	0.057(0.008)	0.482(0.006)	0.252(0.008)	0.283(0.012)	0.348(0.008)
$a = 1.0$	0.009(0.001)	0.007(0.001)	0.009(0.001)	0.444(0.007)	0.197(0.007)	0.194(0.008)	0.307(0.005)
$a = 2.0$	0.002(0.000)	0.002(0.000)	0.003(0.000)	0.394(0.008)	0.191(0.005)	0.194(0.005)	0.305(0.004)
				$b = 0.30$			
$a = 0.5$	0.016(0.001)	0.016(0.001)	0.076(0.014)	0.430(0.008)	0.174(0.008)	0.188(0.011)	0.286(0.008)
$a = 1.0$	0.002(0.000)	0.001(0.000)	0.006(0.004)	0.371(0.011)	0.133(0.006)	0.136(0.007)	0.258(0.006)
$a = 2.0$	0.000(0.000)	0.000(0.000)	0.000(0.000)	0.276(0.009)	0.129(0.005)	0.130(0.005)	0.256(0.005)

Table 3: The means (and standard errors) of the misclassification rate for Simulation 3 over 50 simulations.

indicates that AKM is capable of leveraging the information about difference between block variances to detect meaningful structures, whereas KM, PL, SBC, and HSC are adversely affected by difference between block variances.

Comparing the results in Table 3 to those in Table 2, we see that the additional difference between the means in different blocks significantly benefits KM, PL, SBC, and HSC, which is expected because many biclustering methods including PL, SBC, and HSC make the explicit assumption that different biclusters should have different means. However, this additional difference between block means also benefits AKM with $\lambda = 0$, $\lambda = 0.1$, and $\lambda = 1$, resulting in even better performance.

Importantly, in Simulation 3, larger b means larger difference between both the means and the variances in different blocks, so there are two kinds of signals present. In this situation, AKM performs much better than KM, PL, SBC, and HSC, indicating the possibility that AKM is suitable for dealing with complex datasets in the real world.

A visual illustration of different biclustering methods is provided in Figure 3, which clearly demonstrates that AKM outperforms other competing biclustering methods.

8 Applications

In this section, we apply our algorithm to three cancer gene expression data sets, all of which were proposed and preprocessed by de Souto et al. [2008]. In all three data sets, the rows represent different samples of tissues, and the columns represent different genes. The samples have already been classified into different groups based on their types of tissue, which means that the true sample cluster labels are available. This enables us to evaluate and compare the performance of our algorithm with four other biclustering algorithms (KM, PL, SBC, and HSC) in terms of sample misclassification rate, which is defined as

$$\text{sample misclassification rate} = \frac{\text{number of samples classified into the wrong cluster}}{\text{total number of samples}}.$$

Smaller sample misclassification rate indicates better performance at clustering samples.

8.1 Breast and Colon Cancer Gene Expression Dataset

The first dataset consists of 104 samples and 182 genes. There are only two types of samples: 62 samples correspond to breast cancer tissues, and 42 samples correspond to colon cancer tissues.

After some initial exploration of the dataset, we found that there are six specific samples (all breast cancer tissues) that contain unusually large expression values in some genes, as shown in Figure 4. When applying AKM to this dataset, our method is not affected by this particular pattern and still partitions samples into two clusters that almost perfectly correspond to the two cancer types. However, when applying the competing biclustering methods including KM, PL, SBC, and HSC to this dataset, they all partition the samples into a cluster that contains the six specific samples and another cluster that contains the rest

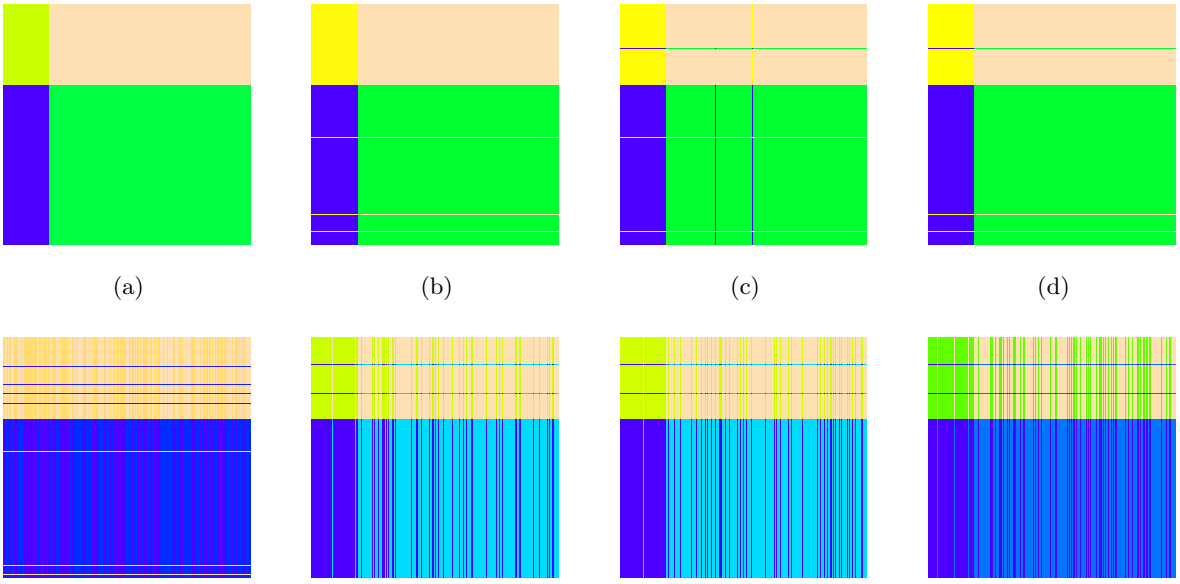


Figure 3: Heatmaps of the estimated mean matrices from different biclustering methods for Simulation 3 with $a = 1.0$ and $b = 0.25$. The rows and columns are reordered so that the true row and column classes form 2×2 contiguous blocks. (a) Ground truth, (b) AKM with $\lambda = 0$, (c) AKM with $\lambda = 0.1$, (d) AKM with $\lambda = 1$, (e) KM, (f) PL, (g) SBC, and (h) HSC.

of the samples. This kind of partition makes sense, because the competing biclustering methods are all essentially based on the assumption that the means in different biclusters are different. However, this kind of partition is still quite different from the breast and colon partition that we desire. Therefore, to ensure a fair comparison, we decide to remove those six specific samples, and the resulting dataset consists of 98 samples and 182 genes.

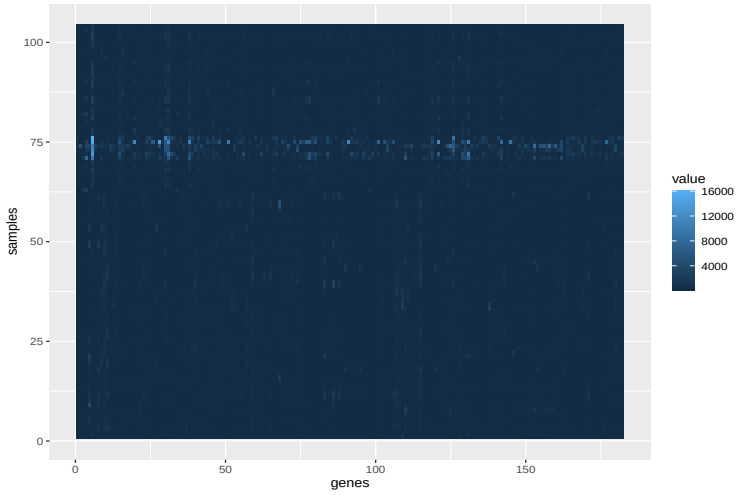


Figure 4: Heatmap of the breast and colon cancer gene expression data matrix. Sample 71 to 76 contain some unusually large expression values.

When applying our biclustering algorithm to real-world data, sometimes we do not have prior knowledge about the appropriate number of biclusters k. In that case, one good way to select k is the "elbow method",

which is also a widely used heuristic method to determine the number of clusters k in traditional k means clustering. The idea is to run the algorithm with $\lambda = 0$ and calculate the loss for different values of k, make a plot with loss on the y-axis and k on the x-axis, and select the k at the point of inflection (the "elbow" of the curve). In Figure 5, we plot the losses for k from 1 to 10 when applying our algorithm with $\lambda = 0$ to the breast and colon cancer dataset. By looking at Figure 5, it is quite clear that our algorithm should select $k = 2$ as the number of biclusters, which is also the true number of row clusters.

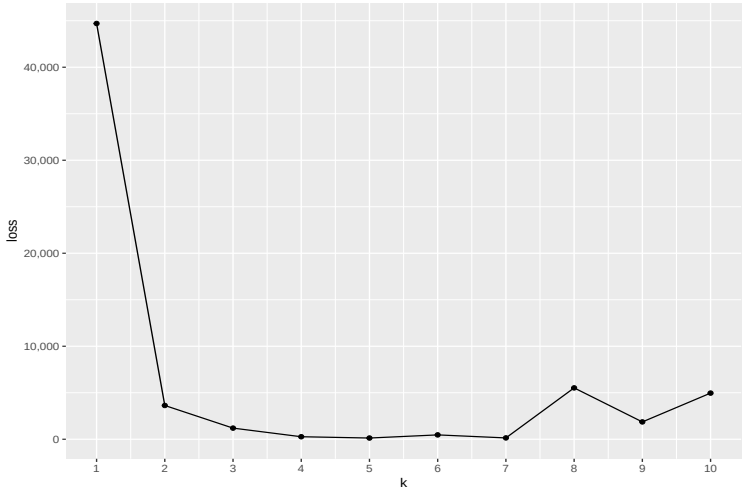


Figure 5: Different losses for k from 1 to 10 on the breast and colon cancer dataset.

Having selected $k = 2$ as the number of biclusters, we apply our algorithm with three different λ values: 0, 0.1 and 1. For comparison, we also apply k-means clustering on the rows (KM), profile likelihood biclustering (PL), sparse biclustering (SBC), and high-order spectral clustering (HSC), with the number of row clusters set to 2. Since PL, SBC, and HSC all allow the number of column clusters to be different from the number of row clusters, we vary the number of column clusters from 1 to 20 and report the best result. For PL, the distribution family is selected as Gaussian. For SBC, the input matrix is always mean-centered before applying the method, and the tuning parameter λ is selected by choosing the λ with the smallest BIC over a grid of λ values, both of which are suggested in their paper. All methods except HSC are run with 100 random initializations.

AKM ($\lambda = 0$)	AKM ($\lambda = 0.1$)	AKM ($\lambda = 1$)	KM	PL	SBC	HSC
0.0306	0.0306	0.0306	0.0816	0.0816	0.0714	0.0816

Table 4: The sample misclassification rates on the breast and colon cancer dataset.

The sample misclassification rates are reported in Table 4. As we can see, our AKM with $\lambda = 0$, 0.1, and 1 achieve the smallest sample misclassification rate of 0.03, whereas other competing biclustering methods have sample misclassification rates ranging from 0.07 to 0.08. Compared to KM which ignores the interaction between samples and genes, our algorithm successfully leverages information about the interaction to significantly improve sample clustering performance.

8.2 Brain Cancer Gene Expression Data Set

The second data set consists of 50 samples and 1739 genes. There are three types of samples: 31, 14, and 5 samples correspond to three different types of brain cancer tissues.

Again, we try to use the elbow method to select the appropriate number of biclusters k for our algorithm. In Figure 6, we plot the losses for k from 1 to 10 when applying our algorithm with $\lambda = 0$ to the brain cancer data set. In this case, it is not completely clear which k we should select, although $k = 2$ or $k = 8$ might be the two most reasonable choices based on the plot alone. However, in some applications we do have prior

knowledge about the appropriate number of biclusters k. For example, we know $k = 3$ should be the default choice for this data set, because three row clusters could be naturally defined based on the three types of samples. Moreover, even if we select $k = 2$ or $k = 8$ based on the plot, the resulting biclusters could reveal interesting findings about the subgroups of samples or genes in this data set.

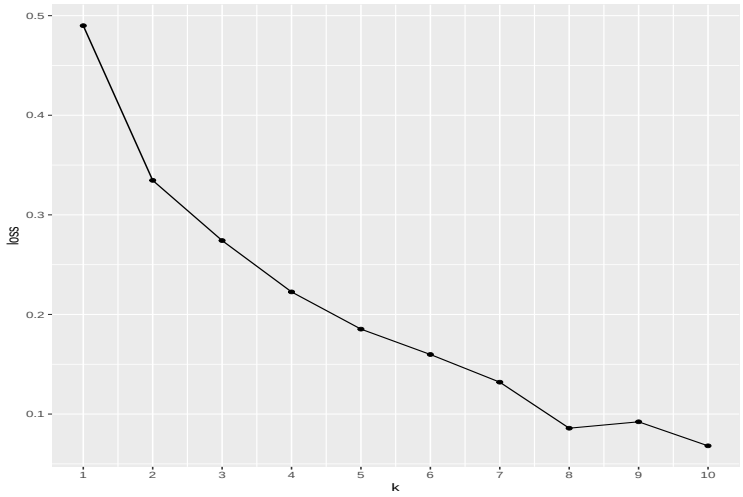


Figure 6: Different losses for k from 1 to 10 on the brain cancer data set.

Having selected $k = 3$ as the number of biclusters, we again apply our algorithm with three different λ values: 0, 0.1 and 1. We also apply KM, PL, SBC, and HSC, with the number of row clusters set to 3. All other settings of the biclustering algorithms are the same as in the first application.

AKM ($\lambda = 0$)	AKM ($\lambda = 0.1$)	AKM ($\lambda = 1$)	KM	PL	SBC	HSC
0.22	0.22	0.22	0.36	0.34	0.34	0.36

Table 5: The sample misclassification rates on the brain cancer data set.

The sample misclassification rates are reported in Table 5. In this case, we again see that our algorithm with all three different values of λ achieve the smallest sample misclassification rate of 0.22, whereas other four biclustering methods all have sample misclassification rates around 0.34. This result indicates that even on larger gene expression data sets with more than two biclusters, our algorithm is still able to significantly outperform other biclustering methods such as KM, PL, SBC, and HSC.

8.3 Prostate Cancer Gene Expression Data Set

The third data set consists of 92 samples and 1288 genes. There are four types of samples: 27 samples correspond to benign prostate tissues, and 13, 32, 20 samples correspond to prostate cancer tissues of three different stages, respectively.

Again, we try to use the elbow method to select the appropriate number of biclusters k for our algorithm. In Figure 7, we plot the losses for k from 1 to 10 when applying our algorithm with $\lambda = 0$ to the prostate cancer data set. In this case, it is clear that our algorithm should select $k = 4$ as the number of biclusters, which is also the true number of row clusters.

Having selected $k = 4$ as the number of biclusters, we again apply our algorithm with three different λ values: 0, 0.1 and 1. We also apply KM, PL, SBC, and HSC, with the number of row clusters set to 4. All other settings of the biclustering algorithms are the same as in the first and second application.

The sample misclassification rates are reported in Table 6. In this case, we see that AKM with $\lambda = 0.1$ and $\lambda = 1$ achieve the smallest sample misclassification rate of 0.4239. AKM with $\lambda = 0$ has a sample misclassification rate of 0.5217, which is slightly worse than PL but better than SBC, KM, and HSC. This

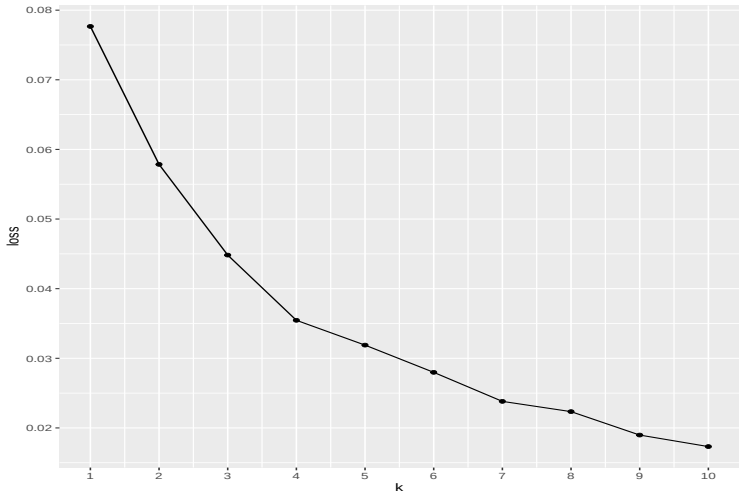


Figure 7: Different losses for k from 1 to 10 on the prostate cancer data set.

AKM ($\lambda = 0$)	AKM ($\lambda = 0.1$)	AKM ($\lambda = 1$)	KM	PL	SBC	HSC
0.5217	0.4239	0.4239	0.5652	0.5109	0.5543	0.5652

Table 6: The sample misclassification rates on the prostate cancer data set.

result once again demonstrates our algorithm's ability to achieve better performance at clustering samples on gene expression data sets compared to other biclustering algorithms such as KM, PL, SBC, and HSC.

9 Discussion

In this paper, we have provided a new formulation of the biclustering problem based on the idea of minimizing the empirical clustering risk. We have developed and proved a consistency result with respect to the empirical clustering risk. Since the optimization problem is combinatorial in nature, finding the global minimum is computationally infeasible. In light of this fact, we have proposed a simple and novel algorithm that finds a local minimum by alternating the use of k-means clustering between columns and rows, and released an R package `akmbiclust` on CRAN that implements the algorithm. We have also provided a probabilistic interpretation of the optimization problem, and proposed extending our method by adding penalization terms. We have evaluated and compared the performance of our algorithm to other related biclustering methods on both simulated data and real-world gene expression data sets. The results have demonstrated that our algorithm is able to detect meaningful structures in the data and outperform other competing biclustering methods in a lot of situations.

One big advantage of our algorithm is its simplicity: the k biclusters can be found simply by applying an adapted version of the k-means clustering algorithm between columns and rows alternately. However, the simplicity comes at the expanse of flexibility: by assigning every row and every column to one and only one bicluster, our method excludes the possibility of overlapping biclusters. Although allowing the biclusters to overlap might be a more reasonable assumption in some cases, we argue that trading off some flexibility for more simplicity is a worthwhile choice for many applications.

In the future, we plan to explore a more general setting of biclustering: biclustering on graphs. The idea is that each column of $\mathbf{X}$ represents a vertex in a graph G, and each row of $\mathbf{X}$ represents a measurement on the vertices. The graph structure of G imposes some restrictions on the column partitions, namely the columns in every bicluster should correspond to vertices in a connected subgraph of G. The problem formulation in Section 2 can be considered as the special case where there is no restriction on the column partitions, and that is equivalent to the graph G being a complete graph with m vertices.

Under this setting of biclustering on graphs, a consistency result could be developed and proved in a

very similar way. The only difference is that the optimization would be over all possible choices of column partitions that "preserve" the graph structure of G. The main theorem still holds true after adding the requirement of $\mathbf{I} \in \mathcal{I}(G)$, where $\mathcal{I}(G)$ denote the set of column partitions that "preserve" the graph structure of G. However, the algorithm presented in Section 5 no longer applies to this setting, and in general people need to either do exhaustive searches over all column partitions in the space of $\mathcal{I}(G)$, or find some heuristic method that could efficiently search through the space of $\mathcal{I}(G)$. This would depend on the specific graph structure of G.

References

A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8:1919–1986, 2007.

A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *Proceedings of the Sixth Annual International Conference on Computational Biology*, pages 49–57, 2002.

S. Bergmann, J. Ihmels, and N. Barkai. Iterative signature algorithm for the analysis of large-scale gene expression data. *Physical Review E*, 67(3):031902, 2003.

G. Biau, L. Devroye, and G. Lugosi. On the performance of clustering in Hilbert spaces. *IEEE Transactions on Information Theory*, 54(2):781–790, 2008.

G. Chen, P. F. Sullivan, and M. R. Kosorok. Biclustering with heterogeneous variance. *Proceedings of the National Academy of Sciences*, 110(30):12253–12258, 2013.

Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, volume 8, page 93, 2000.

E. C. Chi, G. I. Allen, and R. G. Baraniuk. Convex biclustering. *Biometrics*, 73(1):10–19, 2017.

E. C. Chi, B. J. Gaines, W. W. Sun, H. Zhou, and J. Yang. Provable convex co-clustering of tensors. *Journal of Machine Learning Research*, 21:1–58, 2020.

H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, pages 114–125. SIAM, 2004.

M. C. de Souto, I. G. Costa, D. S. de Araujo, T. B. Ludermir, and A. Schliep. Clustering cancer gene expression data: a comparative study. *BMC Bioinformatics*, 9(1):497, 2008.

I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274, 2001.

I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 89–98, 2003.

C. Flynn and P. Perry. Profile likelihood biclustering. *Electronic Journal of Statistics*, 14(1):731–768, 2020.

G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Sciences*, 97(22):12079–12084, 2000.

G. Govaert and M. Nadif. Clustering with block mixture models. *Pattern Recognition*, 36(2):463–473, 2003.

G. Govaert and M. Nadif. An em algorithm for the block mixture model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):643–647, 2005.

G. Govaert and M. Nadif. Block clustering with bernoulli mixture models: Comparison of different approaches. *Computational Statistics & Data Analysis*, 52(6):3233–3245, 2008.

J. Gu and J. S. Liu. Bayesian biclustering of gene expression data. *BMC Genomics*, 9(S4), 2008.

R. Han, Y. Luo, M. Wang, and A. R. Zhang. Exact clustering in tensor block model: Statistical optimality and computational limit. *arXiv preprint arXiv:2012.09996*, 2020.

J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337): 123–129, 1972.

S. Hochreiter, U. Bodenhofer, M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, T. Khamiakova, S. Van Sanden, D. Lin, W. Talloen, et al. FABIA: factor analysis for bicluster acquisition. *Bioinformatics*, 26(12):1520–1527, 2010.

T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, volume 2, pages 688–693, 1999.

Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13(4):703–716, 2003.

L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, pages 61–86, 2002.

M. Lee, H. Shen, J. Z. Huang, and J. S. Marron. Biclustering via sparse singular value decomposition. *Biometrics*, 66(4):1087–1095, 2010.

G. Li, Q. Ma, H. Tang, A. H. Paterson, and Y. Xu. QUBIC: a qualitative biclustering algorithm for analyses of gene expression data. *Nucleic Acids Research*, 37(15):e101, 2009.

T. Linder. Learning-theoretic methods in vector quantization. In *Principles of Nonparametric Learning*, pages 163–210. Springer, 2002.

S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

S. Mankad and G. Michailidis. Biclustering three-dimensional data arrays with plaid models. *Journal of Computational and Graphical Statistics*, 23(4):943–965, 2014.

T. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. In *Pacific Symposium on Biocomputing*, pages 77–88. World Scientific, 2003.

A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17:243–252, 2001.

A. A. Shabalin, V. J. Weigman, C. M. Perou, and A. B. Nobel. Finding large average submatrices in high dimensional data. *The Annals of Applied Statistics*, 3(3):985–1012, 2009.

M. Sill, S. Kaiser, A. Benner, and A. Kopp-Schneider. Robust biclustering by sparse singular value decomposition incorporating stability selection. *Bioinformatics*, 27(15):2089–2097, 2011.

K. M. Tan and D. M. Witten. Sparse biclustering of transposable data. *Journal of Computational and Graphical Statistics*, 23(4):985–1008, 2014.

A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18:136–144, 2002.

A. Tanay, R. Sharan, M. Kupiec, and R. Shamir. Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proceedings of the National Academy of Sciences*, 101(9):2981–2986, 2004.

A. Tanay, R. Sharan, and R. Shamir. Biclustering algorithms: a survey. *Handbook of Computational Molecular Biology*, 9(1-20):122–124, 2005.

C. Tang, L. Zhang, A. Zhang, and M. Ramanathan. Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. In *Proceedings 2nd Annual IEEE International Symposium on Bioinformatics and Bioengineering*, pages 41–48. IEEE, 2001.

H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 394–405, 2002.