Ioannis Liolios

Co-clustering on binary hypergraphs

Master's thesis

September 7th 2025

Thesis supervisor:

Dr. Mirko Signorelli

Leiden University

Mathematical Institute

Abstract

This thesis investigates co-clustering in binary hypergraphs as a means to reduce complexity and improve interpretability in higher-order network data. We adapt a variational expectation–maximization (VEM) framework to jointly cluster nodes and hyperedges, providing a parsimonious representation of latent structure. Through simulation studies, we show that the method recovers parameters with high accuracy when group differences are pronounced, while closer parameter settings require larger sample sizes for reliable performance.

Contents

1 Introduction

Networks provide a fundamental framework for modeling and analyzing many real-world systems, including transportation grids and biological processes. These systems are often represented using graphs, which are mathematical structures composed of vertices (nodes) and edges that capture pairwise relationships. Graph-based models have been studied extensively, which has helped with understanding the behavior of a variety of complex networks (Newman, 2003). However, as the systems we study become increasingly intricate, the limitations of these tools become apparent.

A primary drawback of graphs is their inability to describe multi-way interactions, which are essential in many domains. For instance, co-authorship networks, ecological interactions and e-mail networks often involve multiple entities interacting simultaneously. In a co-authorship network, for example, a graph would represent a paper co-authored by authors A, B, and C as three pairwise edges (A–B, B–C, A–C), losing the information that all three collaborated on the same paper. To capture these higher order interactions, hypergraphs extend the graph framework by introducing hyperedges, a generalization of edges that connect any number of nodes at once. This added flexibility has made hypergraphs a powerful tool for modeling complex systems (Battiston et al., 2020).

Hypergraphs often exhibit some underlying structure which is not directly observable. For instance, consider a hypergraph in which vertices are concert-goers, and the hyperedges are the set of people who attended a particular concert. The genre of music being played or the individual preferences of attendees directly influence the resulting hypergraph. If this structure is not known or observable, it is referred to as latent. Modeling the latent structure of hypergraphs is therefore of great interest, as it can reveal deeper mechanisms governing how interactions are formed. Ng and Murphy (2022) focused on modelling the latent structure of the hyperedges, by clustering the hyperedges based on which nodes participated in them. On the other hand, Brusa and Matias (2024) addressed the latent structure of the nodes, by clustering the nodes into communities with similar behaviours. Simultaneous clustering of both nodes and hyperedges, known as co-clustering, remains relatively unexplored.

In this work, we develop a co-clustering approach for hypergraphs that simultaneously models latent group structures of both nodes and hyperedges. An Expectation-Maximization (EM) algorithm is first introduced to clarify the estimation procedure, but due to its intractability, we proceed with a Variational EM (VEM) approach that makes estimation feasible.

1.1 Graphs

A network can be represented by a graph, which consists of nodes and the connections between them, known as edges. Graphs provide a framework for modeling systems with pairwise interactions, and are formally defined as follows:

Definition 1.1 (Graph)**.** *A graph G is the ordered pair $G = (V, E)$, where $V = \{v_1, v_2, \dots v_N\}$ is a set of N vertices (or nodes) and $E \subseteq \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j\}$ is the set of M edges, each defined as a pair of distinct vertices.*

If the edge (v_i, v_j) is an *ordered* pair, then the graph is called *directed*. In a directed graph, an interaction is determined not only by the nodes involved but also by their ordering, so (v_i, v_j) and (v_j, v_i) are distinct edges. If instead (v_i, v_j) is an *unordered* pair, the graph is called *undirected*. In this case, an interaction is fully specified by the participating nodes, and (v_i, v_j) and (v_j, v_i) represent the same edge. A common representation of a graph is through its adjacency matrix, $\boldsymbol{X}$. The adjacency matrix is a $N \times N$ collection of indicator functions denoting whether an interaction between two elements exists. Formally,

$$x_{ij} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E \\ 0, & \text{if } (v_i, v_j) \notin E. \end{cases}$$

Note that self-interactions are prohibited in this formulation, as $x_{ii} = 0$ by definition. This restriction is meaningful in certain contexts, such as in a friendship network, where self-interactions (e.g., someone being friends with themselves) are not considered. However, in situations where self-interactions are relevant, i.e. when $x_{ii} = 1$ for some node i, the graph would include self-loops. A graph that explicitly excludes self-loops is referred to as a *simple* graph.

Another extension is that of *weighted* graphs. A weighted graph contains information not only about which interactions occur, but also about the magnitude (weight) of that interaction. These extensions can be found in supply chain networks (Arora and Ventresca, 2018), where each edge represents a flow of goods, with the weight of the edge representing the amount of goods being transported. If such information is not included, then we have an *unweighted* graph.

Despite the flexibility that graphs offer, they suffer from one major drawback. While they can represent multiple types of pairwise interactions, they are not capable of modeling higher-order interactions. For example, let us consider a co-authorship network in which the authors A, B and C all share a pairwise edge between each other. It is unclear whether three co-authorships took place (A with B, B with C and C with A), or just a single one (A, B and C all together), as those yield identical representations in a graph. This scenario is illustrated in Figure 1. Hypergraphs are a natural extension of graphs which allow for this specification of the participating nodes of the higher-order interactions.

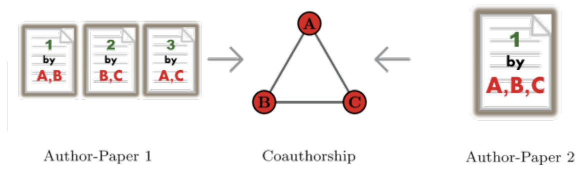

Figure 1: A visualization of the coauthorship problem. In the first scenario, one paper is coauthored by authors A-B, one paper is coauthored by B-C, and one paper is coauthored by A-C. In the second scenario, a single paper is jointly coauthored by A, B and C. In the graph representation, both scenarios result in the same graph representation.

1.2 Hypergraphs

Definition 1.2 (Hypergraph)**.** *A hypergraph is an ordered pair $G = (V, E)$, where $V = \{v_1, \dots, v_N\}$ is the set of N vertices, and*
$E = \{e_j = (v_{j_1}, \dots, v_{j_s}) | v_{j_1}, \dots, v_{j_s}$ have jointly interacted, $j = 1, \dots, M\}$ is the set of M hyperedges.

The index s within each hyperedge denotes the size of the hyperedge, i.e. the number of nodes that interacted. The hyperedge set E is a subset of all the possible combinations of the node set, V. Thus, we can see that $E \subseteq \mathcal{P}(V)$, where $\mathcal{P}$ is the power set. Given that the power set of a set of size N has cardinality equal to 2^N, it follows that $M \leq 2^N$. While graphs typically use an adjacency matrix to represent pairwise connections, this does not extend naturally to higher-order interactions. To accommodate these, the incidence matrix is used, which is a generalization of the adjacency matrix that encodes higher-order interactions. It is a $M \times N$ matrix where the rows represent the hyperedges, while the columns represent the nodes. The entry in the j-th row and the i-th column is equal to 1 if the i-th node participates in the j-th hyperedge, and 0 otherwise.

$$x_{ji} = \begin{cases} 1, & \text{if } v_i \in e_j \\ 0, & \text{if } v_i \notin e_j \end{cases}$$

To aid in understanding this notation, we consider the following incidence matrix with 7 nodes and 3 hyperedges.

$$X = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{1}$$

In equation (1), the first hyperedge is $e_1 = (1, 2, 3)$, the second hyperedge is $e_2 = (4, 5, 6)$, and the last hyperedge is $e_3 = (1, 6, 7)$. We can visualize hypergraphs as overlapping subsets of the node set, similarly to a Venn diagram. In our case, the visualization of the hypergraph is in Figure 2.

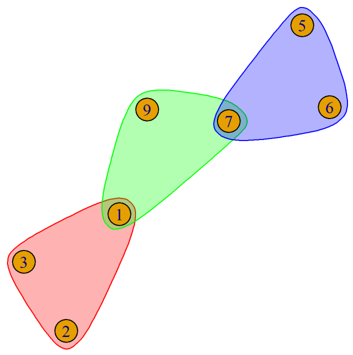

Figure 2: The visualization of the hypergraph with incidence matrix (1). Each colored shape represents a hypergraph, while the orange circles represent the individual nodes.

In principle, a single node may appear multiple times in the same hyperedge, and there is no minimum requirement on the number of nodes per hyperedge. If we enforce a maximum of one appearance of each node per hyperedge and consider hyperedges of size $m \geq 2$ (i.e. at least a pairwise interaction), we construct a *simple* hypergraph. In such hypergraphs, each hyperedge can only appear once.

Definition 1.3 (Simple hypergraph)**.** *A hypergraph $G = (V, E)$ is called simple if every node appears at most once per hyperedge, every hyperedge is at least of size 2, and no hyperedge is repeated.*

While the simplicity assumption is common in hypergraph-based models, especially in applications such as social network analysis (Xiao et al., 2024), we do not impose it in this thesis. Hyperedges may be of any size and may appear multiple times. We introduce the notion of simple hypergraphs solely due to their prevalence in the literature. Nevertheless, all derived results apply equally when restricted to simple hypergraphs.

1.3 EM Algorithm

The Expectation-Maximization (EM) algorithm, introduced by Dempster et al. (1977), is an iterative maximum likelihood estimation procedure for statistical models that depend on latent (i.e. unobserved) variables. These can be variables that influence the observed data but can not be directly measured. Whereas maximum likelihood estimation models without latent variables can be straightforward, the same procedure can not be directly applied into latent models. An example is given below.

Definition 1.4 (Mixture distribution)**.** *Let $f_1(x), \ldots, f_K(x)$ be a collection of probability density functions, and let $\boldsymbol{c} = (c_1, \ldots, c_K)$ be a vector of non-negative mixture weights such* *that* $\sum_{k=1}^{K} c_k = 1$. *A random variable* X *is said to follow a* mixture distribution *with component densities* $f_1, \ldots, f_k$ *and weights* $\boldsymbol{c}$ *if its probability density function is given by:*

$$f_X(x) = \sum_{k=1}^{K} c_k f_k(x).$$

The mixture distribution reflects a scenario where each observation of a random variable gets generated from density f_k with probability c_k. This can happen when the data is generated by several subpopulations, each with its own parameters, but we only observe the overall result without knowing which subpopulation each observation came from.

Assume we have a random variable X which follows a Gaussian mixture distribution. This means, that for mixture weights $c_1, \ldots, c_K$ and mixture parameters $\theta_1 = (\mu_1, \sigma_1), \ldots, \theta_K$, each observation of X comes from $\mathcal{N}(\mu_k, \sigma_k)$ with probability c_k. Let f_k denote the probability density function of the k-th component normal distribution. Let $Z = (Z_1, \ldots, Z_n)$ be the latent unobserved random variable of group assignments for the random vector $(X_1, \ldots, X_n)$. For the $i-th$ observation X_i, $Z_i = k$ if-f the observation was generated by the $k-th$ component. This information is hidden, hence the characterization "latent". The density of each observation X_i under the mixture is

$$\begin{aligned}\mathbb{P}(X_i = x_i) &= \sum_{k=1}^{K} c_k f_k(x_i) \\ &= \sum_{k=1}^{K} c_k \frac{1}{\sqrt{2\pi}\,\sigma_k} \exp\left(-\frac{(x_i - \mu_k)^2}{2\sigma_k^2}\right), \end{aligned} \tag{2}$$

Using $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ and $\boldsymbol{c} = (c_1, \ldots, c_k)$, the marginal log-likelihood of the entire vector can be expanded as follows

$$\begin{aligned} logL(\boldsymbol{\theta}, \boldsymbol{c}|X) &= log(\mathbb{P}(X_1 = x_1, \ldots, X_n = x_n)) \\ &= log\left(\prod_{i=1}^{n} \mathbb{P}(X_i = x_i)\right) \\ &= \sum_{i=1}^{n} log(\mathbb{P}(X_i = x_i)) \\ &= \sum_{i=1}^{n} log\left(\sum_{k=1}^{K} c_k f_k(x_i)\right) \\ &= \sum_{i=1}^{n} log\left(\sum_{k=1}^{K} c_k \frac{1}{\sqrt{2\pi}\,\sigma_k} \exp\left(-\frac{(x_i - \mu_k)^2}{2\sigma_k^2}\right)\right). \end{aligned} \tag{3}$$

Direct optimization of the likelihood in (3) w.r.t. μ_k, σ_k, and c_k is infeasible because the group assignments Z are unobserved. As a result, the likelihood involves a sum over all possible group assignments, leading to a complicated, nonconvex surface. However, if we had access to information about either the true parameter values or the exact component that generates each observation, an analytical solution could be obtained. We present these two situations.

If the component assignment of each observation was known, we could perform MLE in the classical way. For example, assume that $Z = (Z_1, \ldots, Z_n)$ is known to us. For each component k, we denote the set of observations generated by this component as

$$I_k = \{i : Z_i = k\}.$$

Let the size of this set be denoted by $n_k = |I_k|$. Since all of the observations whose index is in I_k are generated from the same distribution, they represent an independent and identically distributed (i.i.d) sample of length n_k from a normal distribution with unknown mean μ_k and unknown variance σ_k^2. We define the marginal likelihood exclusively for the observations that were generated by the k-th component as $L^{(k)}$, and its respective marginal log-likelihood as $l^{(k)}$. We can decompose the model's full marginal likelihood, $L(\boldsymbol{\theta}|X)$, as

$$L(\boldsymbol{\theta}|X) = \prod_{k=1}^{K} L^{(k)}(\theta_k|X).$$

Similarly, we decompose the marginal log-likelihood, $l(\boldsymbol{\theta}|X)$ as

$$l(\boldsymbol{\theta}|X) = \sum_{i=1}^{K} l^{(k)}(\theta_k|X).$$

This follows from the distinctness of the elements of the sets I_k. Each component specific marginal log-likelihood, $l^{(k)}$, can be calculated

$$\begin{aligned}
l^{(k)}(\theta_k|X) = logL^{(k)}(\theta_k|X) &= log\left(\prod_{i\in I_k} f_k(X_i)\right) \\
&= log\left(\prod_{i\in I_k} \frac{1}{\sqrt{2\pi}\,\sigma_k} \exp\left(-\frac{(X_i-\mu_k)^2}{2\sigma_k^2}\right)\right) \\
&= \sum_{i\in I_k} log\left(\frac{1}{\sqrt{2\pi}\,\sigma_k} \exp\left(-\frac{(X_i-\mu_k)^2}{2\sigma_k^2}\right)\right) \\
&= -\frac{n_k}{2}\log(2\pi) - n_k \log\sigma_k - \frac{1}{2\sigma_k^2}\sum_{i\in I_k}(X_i-\mu_k)^2.
\end{aligned}$$

Maximization with respect to μ_k and σ_k respectively yields the maximum likelihood estimates $\hat{\mu}_k, \hat{\sigma}_k$

$$\hat{\mu}_k = \overline{X} = \frac{1}{n_k} \sum_{i \in I_k} X_i,$$
$$\hat{\sigma}_k = \frac{1}{n_k} \sum_{i \in I_k} (X_i - \overline{X})^2.$$

This process can be repeated independently for every k, providing maximum likelihood estimates for every parameter of interest.
Inversely, we assume that the true parameter values were known, but the group assignment variable $Z = (Z_1, \ldots, Z_n)$ remained latent. We first give the definition of the posterior distribution of the latent variable Z.

Definition 1.5 (Posterior distribution of the latent variable). *Let $X = (X_1, \ldots, X_n)$ be an observation vector from a mixture distribution with K components. Let $Z = (Z_1, \ldots, Z_n)$ denote the latent group assignment variable. The quantity $\mathbb{P}(Z|X)$ is called the posterior distribution of the latent variable.*

The posterior distribution of the latent variable is the probability that an observation was generated from a certain component. We can expand it using Bayes' formula

$$\begin{aligned}\mathbb{P}(Z_i = k|X_i) &= \frac{\mathbb{P}(Z_i = k) f_k(X_i)}{f(X_i)} \\ &= \frac{c_k f_k(X_i)}{\sum\limits_{k'=1}^{K} c_{k'} f_{k'}(X_i)} \\ &= \frac{c_k \frac{1}{\sqrt{2\pi}\,\sigma_k} \exp\left(-\frac{(X_i - \mu_k)^2}{2\sigma_k^2}\right)}{\sum\limits_{k'=1}^{K} c_{k'} \frac{1}{\sqrt{2\pi}\,\sigma_{k'}} \exp\left(-\frac{(X_i - \mu_{k'})^2}{2\sigma_{k'}^2}\right)}. \end{aligned} \tag{4}$$

Since all of the parameters used in (4) are known, estimating $\mathbb{P}(Z_i = k|X_i)$ only requires a straightforward computation.
Thus, if we knew the latent group assignments, we could estimate each component's parameters. Similarly, if we knew each component's parameters, we could estimate the latent group assignments. The EM algorithm builds on top of these observations. Before we present it, we introduce the complete data likelihood, which is the likelihood with the assumption that Z is known.

Definition 1.6 (Complete data likelihood)**.** *Let $X = (X_1, \dots, X_n)$ be an observation vector generated by a k-component mixture. The complete data likelihood $L_c(\boldsymbol{\theta}, \boldsymbol{c}|X, Z)$ and the complete data log-likelihood $l_c(\boldsymbol{\theta}, \boldsymbol{c}|X, Z)$ are*

$$L_c(\boldsymbol{\theta}, \boldsymbol{c}|X, Z) = \prod_{i=1}^{N}\prod_{k=1}^{K} \left(c_k f_k(X_i; \theta_k)\right)^{Z_i} \tag{5}$$

and

$$l_c(\boldsymbol{\theta}, \boldsymbol{c}|X, Z) = logL_c(\boldsymbol{\theta}, \boldsymbol{c}|X, Z) = \sum_{i=1}^{N}\sum_{k=1}^{K} Z_i log\left(c_k f_k(X_i; \theta_k)\right) \tag{6}$$

As the latent group assignments are not actually known, we work with the expectation of the complete data log likelihood w.r.t. $Z|X$,

$$\begin{aligned}\mathbb{E}_{Z|X}\left[l_c(\boldsymbol{\theta}, \boldsymbol{c}|X, Z)\right] &= \mathbb{E}_{Z|X}\left[\sum_{i=1}^{N}\sum_{k=1}^{K} Z_i log\left(c_k f_k(X_i; \theta_k)\right)\right] \\ &= \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{Z|X}[Z_i] log\left(c_k f_k(X_i; \theta_k)\right) \\ &= \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{P}(Z_i = k|X) log\left(c_k f_k(X_i; \theta_k)\right).\end{aligned} \tag{7}$$

It is important to distinguish between the complete data likelihood, and the *marginal likelihood* as defined in (3). In terms of notation, $L(\cdot), l(\cdot)$ are used to denote the marginal likelihood and marginal log-likelihood respectively, whereas $L_c(\cdot)$ and $l_c(\cdot)$ are used to denote the complete-data likelihood and complete-data log-likelihood. The complete data likelihood, L_c, assumes that the latent variable Z is observed, and includes that information in the likelihood expression, resulting in a simpler form. In contrast, the marginal likelihood, L, reflects the actual setting where Z is unobservable. It accounts for this uncertainty by summing over all possible values, leading to a more complex expression. Although the marginal likelihood is the true likelihood of interest for inference, it can be problematic to work with directly, as shown in the normal mixture example. The EM addresses this issue by working with the expected value of the complete data likelihood.

In essence, we pretend to know the component-specific parameters to calculate the expectation of the complete data log-likelihood (E-step), and then we use that result to maximize that same likelihood with respect to our parameters of interest (M-step). Dempster et al. (1977) showed that the parameters of the mixture model can be estimated by iterating between the E and M steps multiple times. We now provide a formal description of the EM algorithm.

Let $X = (X_1, \dots, X_n)$ denote the observation vector generated by a mixture distribution with K components. Let the expectation of the complete data log-likelihood be denoted by $Q(\boldsymbol{\theta}, \boldsymbol{c}) = \mathbb{E}_{Z|X}[l_c(\boldsymbol{\theta}, \boldsymbol{c}|X, Z)]$. The EM algorithm is now described as follows:

1. Set $t = 0$

2. Generate random starting estimates of the mixture weights, $\boldsymbol{c}^{(0)}$, and the mixture component parameters, $\boldsymbol{\theta}^{(0)}$.

3. **E-Step**
   Evaluate $Q(\boldsymbol{\theta}, \boldsymbol{c}|\boldsymbol{\theta}^{(t)}, \boldsymbol{c}^{(t)}) = \mathbb{E}_{Z|X}[l_c(\boldsymbol{\theta}^{(t)}, \boldsymbol{c}^{(t)}|X, Z)]$

4. **M-Step**
   Update the mixture weights $\boldsymbol{c}$ and model parameters $\boldsymbol{\theta}$ by maximizing Q

$$\boldsymbol{c}^{(t+1)} = \arg\max_{\boldsymbol{c}} Q(\boldsymbol{\theta}, \boldsymbol{c}|\boldsymbol{\theta}^{(t)}, \boldsymbol{c}^{(t)})$$

   and

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{c}|\boldsymbol{\theta}^{(t)}, \boldsymbol{c}^{(t+1)})$$

5. Increase t by 1

6. Repeat steps 3 to 5 with the new estimates until convergence, or until the maximum number of iterations is reached.

The convergence of the EM algorithm can be determined by two different methods (Wu, 1983). The first approach is to check whether the change in the Q-function between iterations falls below a predetermined threshold ϵ. The other approach is to monitor the change in the parameter estimates and mixture weights, and declare convergence once these changes are smaller than ϵ . In this thesis, we adopt the latter criterion, using the change in parameters and weights to assess convergence. We will claim the EM algorithm has converged when

$$|\theta_k^{(t+1)} - \theta_k^{(t)}| < \epsilon, \tag{8}$$

$$|c_k^{(t+1)} - c_k^{(t)}| < \epsilon, \tag{9}$$

$$\forall k \in \{1, \dots, K\}, \tag{10}$$

for any two consecutive iterations, for some chosen $\epsilon > 0$. If that condition is not satisfied within the predetermined maximum number of iterations, T_{max}, the algorithm is also terminated, but it has not converged.

The EM algorithm has been proven to be correct (Rubin and Little, 1991), meaning that increases in the complete data likelihood, $Q(\boldsymbol{\theta}, \boldsymbol{c}|\boldsymbol{\theta}^{(t)}, \boldsymbol{c}^{(t)})$, guarantee increases in the model's marginal likelihood, $L(\boldsymbol{\theta}^{(t)}, \boldsymbol{c}^{(t)}|X)$. Thus, since $Q(\boldsymbol{\theta}, \boldsymbol{c}|\boldsymbol{\theta}^{(t)}, \boldsymbol{c}^{(t)})$ increases in every iteration, the marginal likelihood also increases. However, this monotonic improvement property can be problematic. Because the EM algorithm guarantees an increase in the expected complete data log-likelihood at each iteration, it is a greedy algorithm. If it is initialized near an underwhelming local maximum, it will converge to that point and cannot escape, since any step away would reduce the objective function. This tendency of EM to get trapped in poor local optima is well studied in the literature (Dempster et al., 1977), and several strategies have been proposed to mitigate it. If prior knowledge about the approximate range or values of the parameters is available, it can be used to guide initialization in a way that increases the chance of converging to a better maximum. In certain cases, such as with a normal mixture model, applying the k-means algorithm can yield parameter estimates that serve as a reasonable starting point, capturing part of the underlying structure of the data (Shireman et al., 2015). When such prior knowledge or techniques are not available, empirical studies suggest that running the algorithm multiple times with different random initializations and selecting the best outcome is an effective strategy (Biernacki et al., 2003).

In some cases, an analytical closed-form update rule for both the E and M steps can be derived. The normal distribution mixture that we demonstrated in one such case. If this derivation is too challenging for either step, a numerical optimization method may be employed. Issues arise when even numerical optimization proves to computationally infeasible, in which case we arrive at a standstill. Multiple extensions of the EM algorithm have been proposed to solve this issue, such as the Classification EM (CEM) and the Variational EM (VEM). The former is employed when we only care about the group assignments, and is a relaxation of the classical EM, whereas the latter uses an approximate distribution to simplify the expression of Q, which it then optimizes.

1.3.1 Classification EM

The Classification EM (Celeux and Govaert, 1985) is a relaxation of the EM presented in the previous section. Instead of using the soft clustering scheme of the EM, we hard cluster each observation into a single group. In this way, a definitive clustering is obtained at each iteration, as opposed to the EM algorithm, which computes a vector of posterior probabilities. The utility of this variant stems from two advantages. Firstly, it is much simpler computationally, and runs significantly faster than a classical EM. Secondly, we estimate the group assignment vector Z in every iteration, making use of the complete data log-likelihood and not the expectation of it. As a result, it can be applied to problems in which the expected complete data likelihood is difficult to compute or optimize. The co-clustering problem of hypergraphs is one of those problems, and we will explore that in more detail in Chapter 2. Let $X = (X_1, \dots, X_n)$ be an observation vector generated by a k-component mixture. The CEM algorithm is described as follows:

1. Set $t = 0$.

2. Generate random starting estimates of the mixture weights, $c_k^{(0)}$, the mixture component parameters, $\theta_k^{(0)}$, and initial class assignments $Z^{(0)}$.

3. **CE-Step (Classification E-Step)**

   For each data point X_i, assign it to the most likely component based on the current parameters:
   $$Z_i^{(t+1)} = \arg\max_k \mathbb{P}(Z_i = k \mid X_i, \boldsymbol{\theta}^{(t)}, \boldsymbol{c}^{(t)}),$$
   i.e., assign each point to the component with the highest posterior probability.

4. **M-Step**

   Update the parameters based on the hard assignments:
   $$\boldsymbol{c}^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{Z_i^{(t+1)} = k\},$$
   and
   $$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} l_c(\boldsymbol{\theta}, \boldsymbol{c}^{(t+1)} \mid X, Z^{(t+1)}),$$
   where l_c is the complete-data log-likelihood.

5. Increase t by 1.

6. Repeat steps 3 to 5 until convergence or until the maximum number of iterations is reached.

1.3.2 Variational EM

An important limitation of EM estimation is that it requires computing the posterior distribution of the latent variable, $\mathbb{P}(Z|X)$. In many cases, this posterior is either analytically intractable or computationally infeasible to estimate. This makes the E-step, which uses $\mathbb{P}(Z|X)$ to evaluate the expected complete-data log-likelihood, difficult or impossible to perform. The Variational EM (Neal and Hinton, 2000) offers a solution by replacing the intractable posterior with a more manageable approximation. The approximating distribution, $q(Z)$, is chosen from a restricted family of distributions typically defined to simplify computation, for example by assuming independence between latent variables. The goal is to find the $q(Z)$ that is closest to the true posterior. Closeness is measured using the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951), which quantifies the difference between two probability distributions:

Definition 1.7 (Kullback-Leibler divergence)**.** *Given distributions Q and P, the Kullback-Leibler (KL) divergence from Q to P is defined as:*

$$D_{\mathrm{KL}}[Q||P] = \int Q \log \frac{Q}{P}$$

The KL divergence is always non-negative and equals zero only when the two distributions are identical (Kullback and Leibler, 1951). A larger KL divergence indicates greater dissimilarity between the distributions, while a smaller value suggests that the distributions are more similar. Thus, the search for a suitable approximation of $\mathbb{P}(Z|X)$ can be framed as minimizing the KL divergence between the true posterior $\mathbb{P}(Z|X)$ and a set of candidate distributions. This leads to the following optimization problem, where the goal is to find the distribution $\hat{q}(Z)$ that minimizes the divergence:

$$\hat{q}(Z) = \arg\min_{q} D_{\mathrm{KL}}[q(Z)||\mathbb{P}(Z|X)].$$

The minimization of the KL divergence is closely related to the maximization of the model's marginal likelihood, through a result known as the *Kullback-Leibler divergence decomposition.*

Lemma 1.1 (Kullback-Leibler divergence decomposition)**.** *Let X be a random variable generated by a k-component mixture. Let Z denote the latent group assignment variable associated with X. Let $q(Z)$ be a distribution over the latent variable. The Kullback-Leibler divergence from $q(Z)$ to $P(Z|X)$ can be decomposed as*

$$D_{KL}[q(Z)||\mathbb{P}(Z|X)] = \mathbb{E}_{q(Z)}[\log q(Z)] - \mathbb{E}_{q(Z)}[\log \mathbb{P}(Z,X)] + \log \mathbb{P}(X)$$

where $\log \mathbb{P}(X)$ *is the marginal log-likelihood.*

Proof.

$$\begin{aligned}
D_{KL}[q(Z)||\mathbb{P}(Z|X)] &= \int_q q(Z) \log \frac{q(Z)}{\mathbb{P}(Z|X)} \\
&= \mathbb{E}_{q(Z)} \left[\log \frac{q(Z)}{\mathbb{P}(Z|X)} \right] \\
&= \mathbb{E}_{q(Z)}[\log q(Z) - \log \mathbb{P}(Z|X)] \\
&= \mathbb{E}_{q(Z)}[\log q(Z)] - \mathbb{E}_{q(Z)}[\log \mathbb{P}(Z|X)] \\
&= \mathbb{E}_{q(Z)}[\log q(Z)] - \mathbb{E}_{q(Z)}[\log \frac{\mathbb{P}(Z,X)}{\mathbb{P}(X)}] \\
&= \mathbb{E}_{q(Z)}[\log q(Z)] - \mathbb{E}_{q(Z)}[\log \mathbb{P}(Z,X)] + \log \mathbb{P}(X).
\end{aligned}$$

□

Re-arranging the terms yields

$$\log \mathbb{P}(X) = \mathbb{E}_{q(Z)}[\log \mathbb{P}(Z,X)] - \mathbb{E}_{q(Z)}[\log q(Z)] + D_{KL}[q(Z)||\mathbb{P}(Z|X)]. \tag{11}$$

The first two terms of the right-hand side constitute the *Evidence Lower Bound* (ELBO) of the variational distribution $q(Z)$, namely

$$\begin{gathered}
ELBO(q(Z)) = \mathbb{E}_{q(Z)}[\log \mathbb{P}(Z,X)] - \mathbb{E}_{q(Z)}[\log q(Z)] \\
\iff \\
\log \mathbb{P}(X) = ELBO(q(Z)) + D_{KL}[q(Z)||\mathbb{P}(Z|X)
\end{gathered} \tag{12}$$

There are two main corollaries of this result. First, because the KL divergence is always non-negative (Kullback and Leibler, 1951) the marginal log-likelihood is always larger than the ELBO, i.e.

$$\log \mathbb{P}(X) \geq ELBO(q(Z)) \tag{13}$$

which establishes the ELBO as a lower bound for the marginal log-likelihood. Second, because $\log \mathbb{P}(X)$ does not depend on the choice of $q(Z)$, the left-hand side of (12) remains constant under the minimization of the KL divergence w.r.t. $q(Z)$. As a result, the right hand side must also remain constant. Therefore, any decrease in the KL divergence corresponds to an equal increase in the ELBO. This implies that minimizing the KL divergence is equivalent to maximizing the ELBO.

Building on these observations, the Variational EM (VEM) algorithm optimizes a lower bound on the marginal likelihood by performing EM updates on the ELBO instead of the complete-data log-likelihood. The E-step is referred to as the *Variational E-step* to emphasize this adjustment. This yields an increasing sequence of lower bounds, which becomes tight when the variational approximation $q(Z)$ closely matches the true posterior.

The VEM algorithm is described as follows:

1. Set $t = 0$.
2. Generate random starting estimates of the mixture weights, $c_k^{(0)}$, the mixture component parameters, $\theta_k^{(0)}$, and the variational parameters $q^{(0)}(Z)$.
3. **VE-Step (Variational E-Step)**
   Update the variational distribution by solving:
   $$q^{(t+1)}(Z) = \arg\max_{q^{(t)}(Z)} \mathbb{E}_{q^{(t)}(Z)}\big[\log \mathbb{P}(X, Z \mid \boldsymbol{\theta}^{(t)}, \boldsymbol{c}^{(t)})\big] - \mathbb{E}_{q^{(t)}(Z)}[\log q^{(t)}(Z)]$$
4. **M-Step**
   Update the parameters by:
   $$\boldsymbol{c}^{(t+1)} = \arg\max_{\boldsymbol{c}} \mathbb{E}_{q^{(t+1)}(Z)}\big[\log \mathbb{P}(X, Z \mid \boldsymbol{\theta}^{(t)}, \boldsymbol{c}^{(t)})\big] - \mathbb{E}_{q^{(t+1)}(Z)}[\log q^{(t+1)}(Z)]$$
   $$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{q^{(t+1)}(Z)}\big[\log \mathbb{P}(X, Z \mid \boldsymbol{\theta}^{(t)}, \boldsymbol{c}^{(t+1)})\big] - \mathbb{E}_{q^{(t+1)}(Z)}[\log q^{(t+1)}(Z)]$$
5. Increase t by 1.
6. Repeat steps 3–5 until convergence or the maximum number of iterations is reached.

Convergence is assessed in the same way as in the standard EM algorithm.
One advantage of the variational approach is that it can mitigate issues such as dependencies between latent group assignments. It has been successfully applied in various settings, including co-clustering problems (Keribin et al., 2012). However, the method yields only an approximate solution, and the quality of this approximation cannot be quantified. This is because evaluating the KL divergence requires access to exact probability values, which are typically intractable in the VEM setting. Furthermore, like the classical EM algorithm, VEM is sensitive to initialization and prone to convergence to local optima (Blei et al., 2017). For this reason, non-random or carefully designed initialization strategies are often necessary.

2 Model

The aim of this chapter is to provide an overview of previously studied models and to introduce our extension. In Section 2.1, we review the existing models along with their associated estimation methods. Section 2.2 then presents our model in detail, including derivations of both its marginal and complete-data likelihood. Following this, Section 2.3 focuses on the application of the CEM and VEM algorithms to our model, obtaining a re-estimation procedure. Finally, Section 2.4 addresses issues related to identifiability and model selection.

2.1 The Latent Class Analysis model

We are interested in modeling the structure of the incidence matrix, $X \in \{0,1\}^{M\times N}$, of a hypergraph. A naive first step would be to assume that every node has the same probability of joining any hyperedge, meaning that every binary entry X_{ji} depends on a shared single parameter, $0 \leq \theta \leq 1$. This would result in the data generating process

$$X_{ji} \sim Bernoulli(\theta), \forall i, j.$$

The marginal likelihood of this model can be computed and optimized, as it is a collection of MN i.i.d. Bernoullis, making it a standard maximum likelihood estimation problem. However, it lacks the complexity to model hyperedge size heterogeneity which is often present in observed networks. A natural extension is the *saturated hypergraph model*, which models each node-hyperedge interaction individually, replacing the single parameter θ with $M \cdot N$ parameters, θ_{ji}. We denote the θ_{ji} matrix as $\boldsymbol{\theta} = \left(\theta_{ji}\right)_{j=1,\ldots,M;\, i=1,\ldots,N}$ The marginal likelihood of this model is

$$\begin{aligned}L(\boldsymbol{\theta}|X) &= \prod_{j=1}^{M} \mathbb{P}(e_j) \\ &= \prod_{j=1}^{M} \mathbb{P}(X_{j1}, X_{j2}, \dots, X_{jN}) \\ &= \prod_{j=1}^{M} \mathbb{P}(X_{j1}) \dots \mathbb{P}(X_{jN}) \\ &= \prod_{j=1}^{M} \prod_{i=1}^{N} \mathbb{P}(X_{ji}) \\ &= \prod_{j=1}^{M} \prod_{i=1}^{N} \theta_{ji}^{x_{ji}} (1-\theta_{ji})^{(1-x_{ji})}\end{aligned} \tag{14}$$

This model not only involves a very large number of parameters that grows with both M, N, but also from the fact that only a single observation is available to estimate each parameter. To overcome both of these issues, the LCA (Ng and Murphy, 2022) assumes a latent clustering of G categories over the hyperedges. This idea is motivated from the fact that, in many applications, the hyperedges may be categorized into intuitive groups. For instance, in a co-authorship network, the group of a hyperedge may represent the field of the publication. In this model, the event-joining probability depends only on the node and the group of the hyperedge, reducing the number of parameters to G(N + 1) - 1. The group of each hyperedge is randomly drawn from a multinomial distribution with parameter $\boldsymbol{\delta} = (\delta_1, \dots, \delta_G)$. By denoting $Z = (Z_1, \dots, Z_M)$ the latent group assignment vector, we have the following data generation process

$$\begin{aligned}&Z_j \sim \mathit{Multinomial}(\boldsymbol{\delta}),\ \ j = 1, \dots, M \\ &X_{ji}|Z_j \sim \mathit{Bernoulli}(\theta_{Z_{ji}}),\ \ \forall i, j\end{aligned}$$

The marginal likelihood of this model is:

$$\begin{aligned}L(\boldsymbol{\theta}, \boldsymbol{\delta}|X) &= \prod_{j=1}^{M} \mathbb{P}(e_j) \\ &= \prod_{j=1}^{M} \sum_{g=1}^{G} \mathbb{P}(e_j|Z_j = g)\mathbb{P}(Z_j = g) \\ &= \prod_{j=1}^{M} \sum_{g=1}^{G} \delta_g \prod_{i=1}^{N} \theta_{gi}^{x_{ji}} (1-\theta_{gi})^{(1-x_{ji})},\end{aligned} \tag{15}$$

as $\mathbb{P}(Z_j = g) = \delta_g$.

Direct maximum likelihood estimation of this model's marginal likelihood is intractable. Ng and Murphy (2022) overcame this challenge by introducing a re-estimation procedure based on a modified Expectation-Maximization algorithm. Building on this, Brusa and Matias (2024) proposed a similar approach focused exclusively on node clustering in hypergraphs. Our work extends these methods by performing simultaneous clustering on both the nodes and hyperedges.

2.2 Cocluster-LCA

We consider a hypergraph with N nodes and M hyperedges, $X \in \{0,1\}^{N\times M}$. We assume it has two independent latent clustering procedures, one for the nodes and one for the hyperedges. The event joining probability depends only on the groups of the node and the hyperedge. We select a fixed number of clusters for each procedure, namely G for the hyperedges and K for the nodes. Let $Z = (Z_1, \dots, Z_M), W = (W_1, \dots, W_N)$ be the latent cluster assignment variables for the hyperedges and nodes respectively, i.e.,

$$\begin{aligned} Z_j = g &\Leftrightarrow \text{Hyperedge } j \text{ belongs to group } g, \\ W_i = k &\Leftrightarrow \text{Node } i \text{ belongs to group } k. \end{aligned}$$

The a priori group assignment probabilities are denoted by $\boldsymbol{\delta} = (\delta_1, \dots, \delta_G)$ for the hyperedge groups, and $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_K)$ for the node groups. These denote the probabilities that any hyperedge (or, respectively, node) originated from a certain group, before the information of X is incorporated.

$$\begin{aligned} \mathbb{P}(Z_j = g) &= \delta_g, \;\; j \in \{1, \dots, M\} \\ &\text{with} \sum_{g=1}^{G} \delta_g = 1, \delta_g \geq 0 \\ \mathbb{P}(W_i = k) &= \gamma_k, \;\; i \in \{1, \dots, N\} \\ &\text{with} \sum_{k=1}^{K} \gamma_k = 1, \gamma_k \geq 0. \end{aligned}$$

Our aim is to simplify the LCA model of equation (15) by making the hyperedge-joining probability parameter depend only on the clusters of each node-hyperedge pair. Thus, we require $G \times K$ parameters for each combination:

$$P(X_{ji} | Z_j = g, W_i = k) = \theta_{gk},$$

which results in the following representation of the data-generating mechanism

$$\begin{aligned}
Z_j &\sim Multinomial(\boldsymbol{\delta}),\ \ j = 1, \ldots, M \\
W_i &\sim Multinomial(\boldsymbol{\gamma}),\ \ i = 1, \ldots, N \\
X_{ji}|Z_j, W_i &\sim Bernoulli(\theta_{Z_j W_i}),\ \ \forall i, j.
\end{aligned}$$

The total number of parameters is $KG + (G-1) + (K-1)$. For relatively small choices of K, G compared to N, M, this number of parameters can become much smaller than the parameters of the saturated model, the hyperedge clustering model (15), and the node clustering model (Brusa and Matias, 2024). This reduction results in a more parsimonious model, whose interpretation is simpler. We denote the θ_{gk} matrix as $\boldsymbol{\theta} = \left(\theta_{gk}\right)_{g=1,\ldots,G;\, k=1,\ldots,K}$. The marginal likelihood of the model is

$$\begin{aligned}
L(\boldsymbol{\delta}, \boldsymbol{\gamma}, \boldsymbol{\theta}|X) &= \prod_{j=1}^{M} \mathbb{P}(e_j) \\
&= \prod_{j=1}^{M} \sum_{g=1}^{G} \mathbb{P}(e_j|Z_j = g)\mathbb{P}(Z_j = g) \\
&= \prod_{j=1}^{M} \sum_{g=1}^{G} \mathbb{P}(Z_j = g) \prod_{i=1}^{N} \mathbb{P}(X_{ji}|Z_j = g) \\
&= \prod_{j=1}^{M} \sum_{g=1}^{G} \mathbb{P}(Z_j = g) \prod_{i=1}^{N} \sum_{k=1}^{K} P(X_{ji}|W_i = k, Z_j = g)P(W_i = k) \\
&= \prod_{j=1}^{M} \sum_{g=1}^{G} \delta_g \prod_{i=1}^{N} \sum_{k=1}^{K} \gamma_k \theta_{gk}^{x_{ji}} (1 - \theta_{gk})^{(1-x_{ji})}. \qquad (16)
\end{aligned}$$

This marginal likelihood cannot be maximized in closed form. Numerical optimization also proves to be infeasible for a reasonably sized dataset, as a single computation of the marginal likelihood requires the summation of $K^N \times G^M$ terms. For example, for a hypergraph which contains $M = 50$ hyperedges, $N = 50$ nodes, and $K = G = 2$ latent groups, this corresponds to summing over 2^{100} terms. In order to estimate the parameters, we attempt to make use of the EM algorithm, which requires us to work with the complete data likelihood. By defining the indicator variables corresponding to each node's/hyperedge's group assignment

- Node-specific group assignment indicator: $W_{ik} = \mathbb{1}_{\{W_i = k\}}$,
- Hyperedge-specific group assignment indicator: $Z_{jg} = \mathbb{1}_{\{Z_j = g\}}$,

we can write the complete data likelihood L_C, and the corresponding complete data log-likelihood, l_C as

$$L_C(\boldsymbol{\theta}, \boldsymbol{\delta}, \boldsymbol{\gamma}|X, Z, W) = \prod_{j=1}^{M}\prod_{g=1}^{G} \delta_g^{Z_{jg}} \prod_{n=1}^{N}\prod_{k=1}^{K} \gamma_k^{W_{ik}} \prod_{j=1}^{M}\prod_{g=1}^{G}\prod_{n=1}^{N}\prod_{k=1}^{K} [\theta_{gk}^{x_{ji}}(1-\theta_{gk})^{1-x_{ji}}]^{Z_{jg}W_{ik}} \tag{17}$$

$$\begin{aligned} l_C(\boldsymbol{\theta}, \boldsymbol{\delta}, \boldsymbol{\gamma}|X, Z, W) = &\sum_{j=1}^{M}\sum_{g=1}^{G} Z_{jg} log(\delta_g) + \sum_{n=1}^{N}\sum_{k=1}^{K} W_{ik} log(\gamma_k) \\ &+ \sum_{j=1}^{M}\sum_{g=1}^{G}\sum_{n=1}^{N}\sum_{k=1}^{K} Z_{jg}W_{ik} log(\theta_{gk}^{x_{ji}}(1-\theta_{gk})^{1-x_{ji}}). \end{aligned} \tag{18}$$

Finally, by taking the expectation of the complete data log-likelihood over $Z, W|X$, we derive the Q-function

$$\begin{aligned} \mathbb{E}_{Z,W|X}\left[l_C(\boldsymbol{\theta}, \boldsymbol{\delta}, \boldsymbol{\gamma}|X, Z, W)\right] = &\mathbb{E}_{Z,W|X}\Bigg[\sum_{j=1}^{M}\sum_{g=1}^{G} Z_{jg} log(\delta_g) + \sum_{n=1}^{N}\sum_{k=1}^{K} W_{ik} log(\gamma_k) \\ &+ \sum_{j=1}^{M}\sum_{g=1}^{G}\sum_{n=1}^{N}\sum_{k=1}^{K} Z_{jg}W_{ik} log(\theta_{gk}^{x_{ji}}(1-\theta_{gk})^{1-x_{ji}})\Bigg] \\ = &\sum_{j=1}^{M}\sum_{g=1}^{G} \mathbb{E}_{Z,W|X}[Z_{jg}] log(\delta_g) + \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{Z,W|X}[W_{ik}] log(\gamma_k) \\ &+ \sum_{j=1}^{M}\sum_{g=1}^{G}\sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{Z,W|X}[Z_{jg}W_{ik}] log(\theta_{gk}^{x_{ji}}(1-\theta_{gk})^{1-x_{ji}}) \end{aligned} \tag{19}$$

The specification of the Q-function requires the calculation of the following posterior distributions:

- $\mathbb{E}_{Z,W|X}[Z_{jg}] = \mathbb{P}(Z_j = g|X_{j\cdot})$,
- $\mathbb{E}_{Z,W|X}[W_{ik}] = \mathbb{P}(W_i = k|X_{\cdot i})$,
- $\mathbb{E}_{Z,W|X}[Z_{jg}W_{ik}] = \mathbb{P}(Z_j = g, W_i = k|X)$.

where $X_{\cdot i}, X_{j\cdot}$ denote the i-th column and j-th row of the incidence matrix X. The first two quantities can be obtained by applying Bayes' rule:

$$\mathbb{P}(Z_j = g|X_{j\cdot}) = \frac{\mathbb{P}(X_{j\cdot}|Z_j = g)\mathbb{P}(Z_j = g)}{\mathbb{P}(X_{j\cdot})}$$

$$= \frac{\delta_g \prod_{i=1}^{N} \sum_{k=1}^{K} \gamma_k \theta_{gk}^{x_{ji}} (1-\theta_{gk})^{1-x_{ji}}}{\sum_{g'=1}^{G} \delta_{g'} \prod_{i=1}^{N} \sum_{k=1}^{K} \gamma_k \theta_{g'k}^{x_{ji}} (1-\theta_{g'k})^{1-x_{ji}}}, \tag{20}$$

$$\mathbb{P}(W_i = k|X_{\cdot i}) = \frac{\mathbb{P}(X_{\cdot i}|W_i = k)\mathbb{P}(W_i = k)}{\mathbb{P}(X_{\cdot i})}$$

$$= \frac{\gamma_k \prod_{j=1}^{M} \sum_{g=1}^{G} \delta_g \theta_{gk}^{x_{ji}} (1-\theta_{gk})^{1-x_{ji}}}{\sum_{k'=1}^{K} \gamma_{k'} \prod_{j=1}^{M} \sum_{g=1}^{G} \delta_g \theta_{gk'}^{x_{ji}} (1-\theta_{gk'})^{1-x_{ji}}}. \tag{21}$$

We apply Bayes' rule to the third quantity,

$$\mathbb{P}(Z_j = g, W_i = k|X) = \frac{\mathbb{P}(X|Z_g = g, W_i = k)\mathbb{P}(Z_j = g)\mathbb{P}(W_i = k)}{\mathbb{P}(X)}. \tag{22}$$

The denominator in (22), $\mathbb{P}(X)$, is the model's marginal likelihood as defined in Equation (16). Because this quantity is intractable to compute, estimating $\mathbb{P}(Z_j = g, W_i = k \mid X)$ becomes computationally infeasible. In the context of latent block models, earlier work by Govaert and Nadif (2008) and Keribin et al. (2012) introduced two EM-based approaches that can be adapted to our setting. The first, Block EM (BEM), generalizes the Variational EM algorithm for coclustering by using a variational distribution as an approximation of the posterior distribution. The second, Classification EM (CEM), takes a more direct route by maximizing the complete-data log-likelihood without relying on such approximations. In the next chapter, we explore these methods in detail and examine their properties in the context of our model.

2.3 Estimation

2.3.1 Block EM (BEM)

The intractability of the posterior distribution $\mathbb{P}(Z_j = g, W_i = k|X)$ can be solved with a VEM approach. We search for a variational distribution $q_{Z,W}(Z, W)$ which is the closest approximation to the intractable posterior, in terms of the KL divergence. We restrict our search to variational distributions with the property $q_{Z,W}(Z, W) = q_Z(Z)q_W(W)$, where

q_Z, q_W are variational distributions that only depends on Z, W respectively. In addition, we assume that these two distribution can be further factorized by

$$q_Z(Z) = \prod_{j=1}^{M} q_{Z_j}(Z_j),$$
$$q_W(W) = \prod_{i=1}^{N} q_{W_i}(W_i).$$

These two simplifications are equivalent to assuming that Z and W are a posteriori independent, and that all the Z_j and all the W_i are pairwise independent among themselves. Throughout the thesis we will use the notation

$$d_j(g) := q_{Z_j}(Z_j = g),$$
$$c_i(k) := q_{W_i}(W_i = k).$$

These are the approximate posterior distributions of node-specific latent assignment probability, and the hyperedge specific latent assignment probability. When referring to the entire set of $c_i(k)$ for all i, k, or the entire set of $d_j(g)$, we use the notation $\boldsymbol{c} = \big(c_i(k)\big)_{i=1,\dots,N;\ k=1,\dots,K}$ and $\boldsymbol{d} = \big(d_j(g)\big)_{j=1,\dots,M;\ g=1,\dots,G}$ respectively. These two are the variational distributions that we aim to optimize with the VEM. Govaert and Nadif (2008) proposed an optimization criterion to be optimized, denoted Φ. This criterion coincides with the ELBO of the variational distribution $q_{Z,W}(Z, W)$ with the posterior independence property between Z, W. The proposed criterion is:

$$\Phi(\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\theta}) = \mathbb{E}_{Z,W|X}\left[l_C(\boldsymbol{\theta}, \boldsymbol{\delta}, \boldsymbol{\gamma}|X, Z, W)\right] + H(\boldsymbol{c}) + H(\boldsymbol{d}) \tag{23}$$

where $H(\boldsymbol{c}), H(\boldsymbol{d})$ denotes the entropy of c, d respectively, i.e.

$$H(\boldsymbol{c}) = -\sum_{i=1}^{N}\sum_{k=1}^{K} c_i(k) log(c_i(k)),$$
$$H(\boldsymbol{d}) = -\sum_{j=1}^{M}\sum_{g=1}^{G} d_j(g) log(d_j(g)).$$

Even with this variational approximation, performing a joint maximization of Φ for both $\boldsymbol{c}, \boldsymbol{d}$ analytically (E-step) remains difficult. Consequently, we will use a fixed point iteration for the E-step, which iteratively maximizes Φ w.r.t. $\boldsymbol{c}$ with fixed $\boldsymbol{d}$, and then maximizes it w.r.t. $\boldsymbol{d}$ for fixed $\boldsymbol{c}$. This process is repeated until convergence is met. Putting everything together, we construct a VEM that uses alternating optimization on the VE-step, optimizing Φ w.r.t. $\boldsymbol{c}, \boldsymbol{d}$ first (using the fixed point algorithm), with fixed $\boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{\delta}$ (VE-step), and

optimizing w.r.t. $\boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{\delta}$ second, with fixed $\boldsymbol{c}, \boldsymbol{d}$ (M-step).
The optimization of Φ with respect to $\boldsymbol{c}$, for fixed $\boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{d}$ yields

$$\hat{c}_i(k) = \frac{\gamma_k \prod_{g=1}^{G} \theta_{kg}^{q_{ig}} (1-\theta_{kg})^{b_g - q_{ig}}}{\sum_{k'=1}^{K} \gamma_{k'} \prod_{g=1}^{G} \theta_{k'g}^{q_{ig}} (1-\theta_{k'g})^{b_g - q_{ig}}}, \tag{24}$$

The optimization of Φ with respect to $\boldsymbol{d}$, for fixed $\boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{c}$ yields

$$\hat{d}_j(g) = \frac{\delta_g \prod_{k=1}^{K} \theta_{kg}^{y_{jk}} (1-\theta_{kg})^{s_k - y_{jk}}}{\sum_{g'=1}^{G} \delta_{g'} \prod_{k=1}^{K} \theta_{kg'}^{y_{jk}} (1-\theta_{kg'})^{s_k - y_{jk}}}, \tag{25}$$

where

$$q_{ig} = \sum_{j=1}^{M} d_j(g) x_{ji}, \quad y_{jk} = \sum_{i=1}^{N} c_i(k) x_{ji}, \quad b_g = \sum_{j=1}^{M} d_j(g), \quad s_k = \sum_{i=1}^{N} c_i(k).$$

The optimization of G with respect to $\boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{\delta}$ for fixed $\boldsymbol{c}, \boldsymbol{d}$ yields

$$\hat{\gamma}_k = \frac{\sum_{i=1}^{N} c_i(k)}{N}, \tag{26}$$

$$\hat{\delta}_g = \frac{\sum_{j=1}^{M} d_j(g)}{M}, \tag{27}$$

$$\hat{\theta}_{gk} = \frac{\sum_{j=1}^{M} \sum_{i=1}^{N} x_{ji} d_j(g) c_i(k)}{\sum_{j=1}^{M} \sum_{i=1}^{N} d_j(g) c_i(k)}. \tag{28}$$

The block EM algorithm is as follows:

1. Initialize $\boldsymbol{c}^{(0)}, \boldsymbol{d}^{(0)}, \boldsymbol{\delta}^{(0)}, \boldsymbol{\gamma}^{(0)}, \boldsymbol{\theta}^{(0)}$

2. For $t = 1, \ldots,$

   2.1. **VE-step**
   Estimate $\boldsymbol{c}^{(t+1)}, \boldsymbol{d}^{(t+1)}$ via a fixed point algorithm

i. Initialize $\tilde{\boldsymbol{d}}^{(0)} = \boldsymbol{d}^{(t)}$

For $s = 1, \ldots,$

ii. Calculate $\tilde{c}_i^{(s)}(k)$ given $(\tilde{\boldsymbol{d}}^{(s-1)}, \boldsymbol{\gamma}^{(t)}, \boldsymbol{\delta}^{(t)}, \boldsymbol{\theta}^{(t)})$ via (24) $\forall i, k$

iii. Calculate $\tilde{d}_j^{(s)}(g)$ given $(\tilde{\boldsymbol{c}}^{(s)}, \boldsymbol{\gamma}^{(t)}, \boldsymbol{\delta}^{(t)}, \boldsymbol{\theta}^{(t)})$ via (25) $\forall j, g$

iv. Increase s by 1

v. Repeat steps (ii) - (iv) until convergence

vi. Update $d_j^{(t+1)}(g) = \tilde{d}_j^{(s)}(g)$

Update $c_i^{(t+1)}(k) = \tilde{c}_i^{(s)}(k)$

2.2. **M-step**

Update $\boldsymbol{\delta}^{(t+1)}, \boldsymbol{\gamma}^{(t+1)}, \boldsymbol{\theta}^{(t+1)}$ with formulas (26), (27), (28) using $\boldsymbol{c}^{(t+1)}, \boldsymbol{d}^{(t+1)}$

2.3. Increase t by 1

3. Repeat steps 2.1-2.3 until convergence or until the maximum number of iterations has been reached.

We consider the BEM algorithm to have converged when

$$|\boldsymbol{\gamma}^{(t)} - \boldsymbol{\gamma}^{(t-1)}| < \epsilon,$$
$$|\boldsymbol{\delta}^{(t)} - \boldsymbol{\delta}^{(t-1)}| < \epsilon,$$
$$|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^{(t-1)}| < \epsilon,$$

for a specified tolerance $\epsilon > 0$, i.e. each parameter shifts less than ϵ between two consecutive iterations.

The re-estimation formulas are again obtained through the constrained optimization of the new criterion, and the proofs are in appendix (A).

2.3.2 Classification EM (CEM)

The classification EM assigns a hard clustering in every iteration, instead of the probabilistic clustering by the classical EM. Therefore, it uses the complete data log-likelihood directly and not its expectation w.r.t. the latent grouping variables. This is particularly useful in the hypergraph co-clustering context, as it avoids the need for any variational approximation. The algorithm is as follows:

1. Initialize $\boldsymbol{\delta}^{(0)}, \boldsymbol{\gamma}^{(0)}, \boldsymbol{\theta}^{(0)}$

2. For $t = 1, \dots,$

   2.1. **E-step**
   Update the posterior probabilities $c_i^{(t)}(k), d_j^{(t)}(g)$ via a fixed point algorithm, using (24), (25)

   2.2. Assign each node v_i and each hyperedge e_j, to the cluster which provides the maximum posterior probability.

$$W_i^{(t+1)} = \arg\max_k (c_i^{(t)}(k)),$$
$$Z_j^{(t+1)} = \arg\max_g (d_j^{(t)}(g)),$$

   2.3. **M-step**
   Update estimates as:

$$\gamma_k^{(t+1)} = \frac{\text{\# of nodes in cluster } k}{N} = \frac{\sum_{i=1}^{N} W_{ik}^{(t+1)}}{N}$$
$$\delta_g^{(t+1)} = \frac{\text{\# of hyperedges in cluster } g}{M} = \frac{\sum_{j=1}^{M} Z_{jg}^{(t+1)}}{M}$$
$$\theta_{gk}^{(t+1)} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} W_{ik}^{(t+1)} Z_{jg}^{(t+1)} X_{ji}}{\sum_{i=1}^{N}\sum_{j=1}^{M} W_{ik}^{(t+1)} Z_{jg}^{(t+1)}}$$

   2.4. Increase t by 1

   2.5. Repeat until convergence or until the maximum number of iterations is reached.

We consider the CEM algorithm to have converged when

$$|\boldsymbol{\gamma}^{(t)} - \boldsymbol{\gamma}^{(t-1)}| < \epsilon,$$
$$|\boldsymbol{\delta}^{(t)} - \boldsymbol{\delta}^{(t-1)}| < \epsilon,$$
$$|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^{(t-1)}| < \epsilon,$$

for a specified tolerance $\epsilon > 0$, i.e. each parameter shifts less than ϵ between two consecutive iterations.

2.4 Model selection

While the Block EM algorithm takes on the task of estimating every parameter of the model, two hyperparameters require our specification. Specifically, these are the number of node groups, K, and the number of hyperedge groups, G. As their true sizes are latent, we require a way to establish a set of candidate models, and then order them based on some measure. Measures such as the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC) are commonly used for such model selection problems. For a given model m with estimated marginal likelihood $\hat{L}$, a parameter set of size k and a sample size of n, they are defined as:

$$BIC(m) = k \cdot \ln(n) - 2\ln(\hat{L})$$
$$AIC(m) = 2k - 2\ln(\hat{L})$$

However, in the case of Cocluster-LCA, the calculation of the marginal likelihood $\hat{L}$ is intractable, making both measures unusable. A potential approach to tackling this issue would be to replace the real marginal likelihood with its variational approximation, through the ELBO. However, the tightness of the bound between the real and variational likelihood is unknown, making this an approximation of uncertain quality. To counter this problem, Biernacki et al. (2000) introduced the Integrated Complete Likelihood (ICL).

2.4.1 ICL

The ICL is defined as the logarithm of the integrated complete likelihood. It is defined as

Definition 2.1 (Integrated Complete Likelihood). *Given a Cocluster-LCA model, R with parameter set $\Omega = (\boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\theta})$, the Integrated Complete Likelihood is*

$$ICL(R) = \int p(X, Z, W|\Omega; M)p(\Omega; M)d\Omega$$

The practicality of the ICL also lies in the ability to easily estimate it. Keribin et al. (2012) have provided a closed-form solution for the ICL of a binary co-clustering model, which is:

Theorem 1 (Closed form solution of the ICL (Keribin et al., 2012)). *Given a Cocluster-LCA model, R, with parameter set $\Omega = (\boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\theta})$, the Integrated Complete Likelihood can be* *computed in closed form as*

$$\begin{aligned} ICL(R) &= \log\Gamma(G) + \log\Gamma(M) - \log\Gamma(N+G) - \log\Gamma(K+M) \\ &+ \sum_k \log\Gamma(N_k+1) + \sum_\ell \log\Gamma(N^g+1) + \sum_{k,g} \log\Gamma(n_k^g+1) \\ &+ \log\Gamma(N_k N^g - n_k^g + 1) - \log\Gamma(N_k N^g + 2) \end{aligned}$$

where $\Gamma(\cdot)$ *is the Gamma function,* K *is the number of node clusters,* G *is the number of hyperedge clusters,* M *is the number of observed hyperedges,* N *is the number of nodes and*

$$N_k = \sum_{i=1}^{N} W_{ik}, \quad N^g = \sum_{j=1}^{M} Z_{jg}, \quad n_k^g = \sum_{i=1}^{N}\sum_{j=1}^{M} W_{ik} Z_{jg} X_{ji}.$$

Z_{jg} and W_{ik} are obtained through the hard clustering of the nodes and hyperedges with the maximization of $d_j(g)$ and $c_i(k)$, respectively.

2.4.2 Selection of K, G

A plethora search algorithms can be employed to search through the candidate models, such as the one developed by Keribin et al. (2012). However, they come at a cost of information loss, as not all possible configurations are examined. Therefore, we opt to use a brute force method of exhausting every single predefined combination of K, G. To this end, we upper and lower bound K, G on arbitrary but computationally feasible values, and fit a model for every combination that falls within the permissible range. Then a ICL value is calculated for each model, and the model with the highest ICL is chosen.

2.4.3 Identifiability and the trace maximization algorithm

The Cocluster-LCA is identifiable up to label permutation under very mild conditions (Brault et al., 2014). These are:

1. $\gamma_k > 0 \quad \forall k \in \{1, \dots, K\}$
2. $\delta_g > 0 \quad \forall g \in \{1, \dots, G\}$
3. $N > 2K - 1$
4. $M > 2G - 1$
5. $K \geq 2$ and $G \geq 2$

6. The matrix product $\boldsymbol{\theta}\boldsymbol{\gamma}^T$ yields a $G \times 1$ vector with distinct entries.

7. The matrix product $\boldsymbol{\delta}\boldsymbol{\theta}$ yields a $1 \times K$ vector with distinct entries.

Conditions 6 and 7 ensure that each latent group corresponds to a distinct distribution. Without these conditions, different parameter configurations could lead to indistinguishable or nearly identical distributions, making the model non-identifiable. Condition 5 is necessary because a Cocluster-LCA model with $K = 1$ or $G = 1$ reduces to a one-dimensional mixture of Bernoulli distributions, which is generally non-identifiable (Gyllenberg et al., 1994). As such, we will restrict the search space for K, G to values greater than or equal to 2.

Because The Cocluster-LCA is identifiable only up to label permutation, any reordering of the group labels results in an equivalent model to the original. For example, assume a model with the simple parameter set $\boldsymbol{c} = (c_1, c_2)$. The following two configurations represent the same underlying model:

$$\boldsymbol{c}^{\boldsymbol{A}} = (c_2, c_1), \quad \boldsymbol{c}^{\boldsymbol{B}} = (c_1, c_2).$$

This label ambiguity is not problematic in settings where the true labels are unknown, such as in real data. However, in simulation studies, where the ground truth is known and performance metrics such as classification accuracy are used, it becomes an important limitation. If the estimated labels are permuted relative to the true labels, accuracy will be artificially low, even when the clustering is otherwise correct. To remedy this, we will use the trace maximization algorithm, which searches for the optimal permutation by maximizing the number of correctly matched labels. We first define the confusion matrix.

For a clustering problem with G groups, let Z^* be the true group labels and $\hat{Z}$ the estimated labels. We define the confusion matrix C as a $G \times G$ matrix where C_{ij} counts the number of samples assigned to group i in Z^* and to group j in $\hat{Z}$. The trace of the confusion matrix, $\mathrm{Tr}(C)$, gives the total number of observations for which the two clusterings have assigned the same label. The trace maximization algorithm aims to maximize the trace of the confusion matrix by permuting the estimated labels. The trace maximization algorithm evaluates all possible label permutations and selects one that maximizes the trace of the confusion matrix between the true and estimated labels. Formally,

1. Generate all $G!$ permutations τ of the G group labels.

2. For each permutation τ, relabel the estimated labels as $\tau(\hat{Z})$, and compute the confusion matrix $C^{(\tau)}$ between Z^* and $\tau(\hat{Z})$.

3. Compute the trace of each confusion matrix, $\mathrm{Tr}(C^{(\tau)})$

4. Select the permutation τ^* that maximizes the trace:

$$\tau^* = \arg\max_{\tau} \mathrm{Tr}(C^{(\tau)}).$$

5. Use $\tau^*(\hat{Z})$ as the relabeled estimated clustering for evaluation.

The trace maximization procedure is applied separately to the hypergraph labels (using $\hat{Z}$ and Z^*) and the node labels (using $\hat{W}$ and W^*). While there is no theoretical guarantee that this algorithm will recover the correct label order, particularly when parameter estimates are poor, our empirical results show that it performs reasonably well in practice.

3 Simulation study

The previous chapter established a modified VEM estimator (referred to as BEM) for the estimation of a Cocluster-LCA model. This chapter complements that theoretical analysis through a series of simulation studies aimed at evaluating the estimator's performance in controlled settings. Simulations allow us to test the estimator in settings where the true parameters are known, the latent structure is fully specified, and certain aspects of the difficulty, such as the level of separation between groups, can be varied.
Our objectives are as follows:

1. To assess the estimator's ability to correctly recover the latent groups of the nodes/hyperedges

2. To measure the accuracy with which it estimates the parameters of group proportions $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ and connecting probabilities $(\boldsymbol{\theta})$.

3. To examine its robustness and scalability as the problem gets more complex

4. To examine the effectiveness of the ICL in selecting the best model

To explore these objectives, we designed 4 Monte Carlo experiments, each constructed to test the estimator under different conditions.
The first two focus on simple coclustering settings with two latent groups. In Experiment 1, the group structure is well-separated, serving as a baseline, while Experiment 2 reduces group separability to challenge the estimator's ability to distinguish similar components. Experiment 3 increases model complexity by introducing three latent groups in both dimensions, with varying degrees of overlap across the group combinations, assessing how performance degrades as the number of groups increases. Finally, Experiment 4 investigates model selection: we simulate data from a three-by-three structure and evaluate whether the ICL criterion correctly identifies this latent configuration when fitting models with varying numbers of groups. To assess scalability, Experiments 1 and 2 are run across different sample sizes, while Experiments 3 and 4 are conducted on a fixed moderate-size dataset. To this end, we implemented the BEM in R and used the ALICE High Performance Computing facility to run the simulations in parallel. The code is available in the GitHub repository `https://github.com/Liolios42/Coclustering-on-binary-hypergraphs`.

Section 3.1 outlines the data generation process, while Section 3.2 introduces the evaluation metrics, and Section 3.3 describes the implementation choices. Section 3.4 presents the results of Simulations 1–3, and Section 3.5 reports the results of Simulation 4. Finally, Section 3.6 discusses the motivation for using multiple starting points.

3.1 Data generating mechanism

In all experiments, we defined the number of nodes N, the number of observed hyperedges M, the number of node/hyperedge groups K, G and their respective prior probabilities $\boldsymbol{\gamma}, \boldsymbol{\delta}$, and the connecting probabilities $\boldsymbol{\theta}$. The mechanism used to simulate the dataset is

1. For each node $v_i, i \in \{1, \dots N\}$ sample its latent group through $W_i \sim Multinomial(\boldsymbol{\gamma})$
2. For each hyperedge $e_j, j \in \{1, \dots M\}$ sample its latent group through $Z_j \sim Multinomial(\boldsymbol{\delta})$
3. For each hyperedge e_j belonging to group g and each node v_i belonging to group k, perform a $Bernoulli(\theta_{gk})$ trial to decide whether the node joins the hyperedge.

This procedure outputs an $M \times N$ matrix X, an $M \times 1$ hyperedge group assignment vector Z^*, and an $N \times 1$ node group assignment vector W^*. The matrix X is used for inference, as it is observable, whereas Z^*, W^* are used for evaluation, as they are latent.

The parameter sets used to generate the datasets are:

Experiment 1:

$$N \in \{200, 1000\}, \quad M \in \{200, 1000\},$$

$$\boldsymbol{\gamma} = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}, \boldsymbol{\delta} = \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix}, \boldsymbol{\theta} = \begin{bmatrix} 0.9 & 0.6 \\ 0.4 & 0.2 \end{bmatrix}.$$

Experiment 2:

$$N \in \{200, 1000\}, \quad M \in \{200, 1000\},$$

$$\boldsymbol{\gamma} = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, \boldsymbol{\delta} = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, \boldsymbol{\theta} = \begin{bmatrix} 0.4 & 0.3 \\ 0.2 & 0.25 \end{bmatrix}.$$

Experiment 3:

$$N = 500, \quad M = 500,$$

$$\boldsymbol{\gamma} = \begin{bmatrix} 0.6 \\ 0.3 \\ 0.1 \end{bmatrix}, \boldsymbol{\delta} = \begin{bmatrix} 0.6 \\ 0.3 \\ 0.1 \end{bmatrix}, \boldsymbol{\theta} = \begin{bmatrix} 0.9 & 0.6 & 0.4 \\ 0.2 & 0.8 & 0.4 \\ 0.2 & 0.7 & 0.1 \end{bmatrix}.$$

Experiment 4:

$$N = 500, \quad M = 500,$$

$$\boldsymbol{\gamma} = \begin{bmatrix} 0.6 \\ 0.3 \\ 0.1 \end{bmatrix}, \boldsymbol{\delta} = \begin{bmatrix} 0.6 \\ 0.3 \\ 0.1 \end{bmatrix}, \boldsymbol{\theta} = \begin{bmatrix} 0.9 & 0.6 & 0.4 \\ 0.2 & 0.8 & 0.4 \\ 0.2 & 0.7 & 0.1 \end{bmatrix}.$$

For clarity, we also assign descriptive names to these experiments: **2x2 easy** (Experiment 1), **2x2 hard** (Experiment 2), **3x3 mixed** (Experiment 3) and **model-selection** (Experiment 4).

3.2 Evaluation metrics

We evaluate the estimator's performance using three metrics: mean absolute error (MAE) of the parameter estimates, classification accuracy, and the adjusted Rand index (ARI). These metrics assess both the quality of parameter recovery and the accuracy of inferred group assignments for nodes and hyperedges.

The mean absolute error is measured using the average elementwise ℓ_1 distance between the estimated and true parameters:

$$l_1(\boldsymbol{\theta}) = \frac{\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_1}{KG},$$
$$l_1(\boldsymbol{\gamma}) = \frac{\|\hat{\boldsymbol{\gamma}} - \boldsymbol{\gamma}\|_1}{K},$$
$$l_1(\boldsymbol{\delta}) = \frac{\|\hat{\boldsymbol{\delta}} - \boldsymbol{\delta}\|_1}{G}.$$

This metric captures how closely the true parameter values are recovered.

Classification accuracy refers to the proportion of nodes or hyperedges whose group labels are correctly identified. For a given node group assignment vector $W^* = (W_1^*, \ldots, W_N^*)$ and hyperedge group assignment vector $Z^* = (Z_1^*, \ldots, Z_M^*)$, let $\hat{W} = (\hat{W}_1, \ldots, \hat{W}_N)$ and $\hat{Z} = (\hat{Z}_1, \ldots, \hat{Z}_M)$ be the respective estimated group assignment vectors. The node and hyperedge classification accuracies are defined as:

$$\textbf{Node Classification Accuracy} = \frac{\sum_{i=1}^{N} \mathbb{1}_{W_i^* = \hat{W}_i}}{N},$$

$$\textbf{Hyperedge Classification Accuracy} = \frac{\sum_{j=1}^{M} \mathbb{1}_{Z_j^* = \hat{Z}_j}}{M},$$

i.e. the number of nodes (or hyperedges) for which the estimated group is the correct one, divided by the total number of nodes (or hyperedges). Because the Cocluster-LCA is a mixture model, the labels of the estimated groups are arbitrary under permutation, i.e. what is estimated as "Node group 1" does not necessarily align with the group we defined as "Node group 1" in the generation of the data. Therefore, a direct comparison between W^* and $\hat{W}$ (or between Z^* and $\hat{Z}$) would be meaningless. To remedy this, we align the estimated labels with the true ones by permuting them via the trace maximization algorithm (as described in Chapter 2.4.3). We apply this algorithm to both $\hat{W}, \hat{Z}$ with reference to W^*, Z^* respectively. The node and hyperedge classification accuracies are then reported after applying this algorithm.

The adjusted Rand index (Rand, 1971) quantifies the similarity between the true and estimated clusterings, correcting for agreement by chance and accounting for arbitrary label permutations. It is particularly appropriate for mixture models, where the group labels are only identifiable up to permutation.

Let $X = \{X_1, \dots, X_n\}$ be n datapoints, and let C_1, C_2 represent two clusterings of X into k distinct clusters. Let a represent the number of pairs of datapoints which are clustered together in C_1, and are clustered together in C_2, and let b represent the number of pairs of datapoints which belong to different clusters in C_1 and different clusters in C_2. For example, we consider the case $k = 2$, $n = 4$. For a sample $X = \{X_1, X_2, X_3, X_4\}$, we consider the clusterings $C_1 = \{(X_1, X_4), (X_2, X_3)\}$, $C_2 = \{(X_1, X_3, X_4), (X_2)\}$. In this case, C_1 clusters X_1, X_4 and X_2, X_3 together, whereas C_2 clusters X_1, X_3, X_4 together, and X_2 separately. The pair (X_1, X_4) is the only one which is clustered together in both C_1 and C_2, which would result in $a = 1$. In constrast, the pairs that are clustered separately in both C_1, C_2 are (X_1, X_2) and (X_4, X_2), resulting in $b = 2$.
The Rand Index is then defined as

$$\text{RI} = \frac{a+b}{\binom{n}{2}}.$$

The Adjusted Rand Index (ARI) adjusts this quantity for the expected value of RI under random labeling:

$$\text{ARI} = \frac{\text{RI} - \mathbb{E}[\text{RI}]}{1 - \mathbb{E}[\text{RI}]},$$

where $\mathbb{E}[\text{RI}]$ denotes the expected RI under chance. This corresponds to the RI obtained if we were to randomly guess each cluster. The calculation of this quantity requires the use of an algorithm found in Vinh et al. (2009).

The ARI takes values between -1 and 1, where 1 indicates perfect agreement between clusterings, 0 corresponds to the expected similarity under random guessing, and negative values indicate less agreement than expected by chance, i.e., systematic disagreement.

3.3 Experimental setup

3.3.1 Simulations 1, 2 and 3

The goal of the first three experiments is to evaluate the accuracy of parameter estimation when the true latent structure is correctly specified. For each experiment, we generated data according to a known configuration of latent groups and ran 500 Monte Carlo replicates. Within each replicate, we used three random initializations on the BEM, and selected the fit with the highest ICL for analysis.

3.3.2 Simulation 4

The goal of the fourth experiment is to assess the ability of the model to recover the correct latent structure when it is not known in advance. In each of the 500 Monte Carlo replicates, we fitted all 9 possible combinations of $K \in \{2, 3, 4\}$ and $G \in \{2, 3, 4\}$ (i.e., 9 models in total). Each model was run with three random initializations, and the solution with the highest ICL across all 27 fits was retained. The corresponding values of (K, G) were then treated as the inferred group structure.

For all experiments, we imposed a limit of 20 BEM iterations, and selected a convergence threshold of $\epsilon = 0.0001$.

3.4 Simulations 1-3 (2x2 easy, 2x2 hard, 3x3 mixed)

Classification Accuracy

When the latent structure was correctly specified, the BEM algorithm consistently achieved high accuracy in both parameter recovery and clustering performance across all settings. A summary table for the accuracies and ARI for each experiment is given in Table 1.

In the 2x2 easy experiment, classification accuracy was almost perfect across all configurations. The model achieved perfect hyperedge classification (100%) in all 500 replicates in 3 out of 4 settings, and maintained nearly perfect performance elsewhere (Table 1). Node classification accuracy was similarly high, exceeding 99.4% in all cases, with ARIs also close to 1, indicating close to perfect recovery of the true partitioning.

Dataset	**N**	**M**	Acc. nodes	Acc. hyperedges	ARI nodes	ARI hyperedges
2x2 easy	200	200	0.998 (0.014)	1.000 (0.000)	0.992 (0.054)	1.000 (0.000)
2x2 easy	200	1000	0.996 (0.040)	0.999 (0.000)	0.987 (0.110)	0.999 (0.002)
2x2 easy	1000	200	0.994 (0.030)	1.000 (0.000)	0.980 (0.111)	1.000 (0.000)
2x2 easy	1000	1000	**0.998 (0.016)**	**1.000 (0.000)**	**0.995 (0.060)**	**1.000 (0.000)**
2x2 hard	200	200	0.852 (0.071)	0.987 (0.009)	0.511 (0.166)	0.952 (0.385)
2x2 hard	200	1000	0.980 (0.085)	0.981 (0.094)	0.951 (0.176)	0.962 (0.189)
2x2 hard	1000	200	0.840 (0.085)	0.991 (0.065)	0.488 (0.187)	0.981 (0.133)
2x2 hard	1000	1000	**0.996 (0.004)**	**0.992 (0.004)**	**0.986 (0.018)**	**0.968 (0.018)**
3x3 mixed	500	500	0.981 (0.094)	0.986 (0.094)	0.984 (0.064)	0.994 (0.042)

Table 1: Classification accuracy and Adjusted Rand Index (ARI) for the 2x2 easy, 2x2 hard, and 3x3 mixed experiments. Values are reported as the mean over 500 Monte Carlo replicates, with standard deviations in parentheses. The best performing model in each experiment is indicated in bold.

The 2x2 hard experiment introduced more overlap between the latent groups, resulting in reduced separability. As expected, this posed a challenge in the small sample case. Specifically, when $M = 200$, the classification accuracy and ARI of the nodes were substantially lower, regardless of the number of nodes. For example, node accuracy was around 85.2% (ARI: 0.511) for $N = 200$, and similarly 84.0% (ARI: 0.488) for $N = 1000$. In contrast, when M was increased to 1000, performance greatly improved: accuracy rose to 99.6% and 98.0%, and ARI to 0.951 and 0.986 for $N = 200$ and $N = 1000$, respectively.

In the 3x3 mixed experiment, the model still performed remarkably well. Node and hyperedge accuracies were 98.1% and 98.6%, respectively, with ARIs above 0.98, showcasing the robustness of the BEM even as the number of latent groups increase.

Estimation error

Parameter estimation errors followed a similar pattern to classification accuracy. We present these results in Table 2. In addition, we present 9 histograms (Figures 3 to 10), which visualize the distribution of estimates for each of the 500 Monte Carlo replicates.

Dataset	**N**	**M**	$\ell_1(\boldsymbol{\gamma})$	$\ell_1(\boldsymbol{\delta})$	$\ell_1(\boldsymbol{\theta})$
2x2 easy	200	200	0.024 (0.021)	0.024 (0.019)	0.004 (0.004)
2x2 easy	200	1000	0.022 (0.023)	0.012 (0.008)	0.002 (0.003)
2x2 easy	1000	200	0.014 (0.027)	0.025 (0.019)	0.003 (0.010)
2x2 easy	1000	1000	**0.013 (0.030)**	**0.011 (0.008)**	**0.001 (0.000)**
2x2 hard	200	200	0.106 (0.088)	0.029 (0.022)	0.009 (0.007)
2x2 hard	200	1000	0.027 (0.080)	0.029 (0.092)	0.005 (0.025)
2x2 hard	1000	200	0.116 (0.101)	0.035 (0.063)	0.011 (0.020)
2x2 hard	1000	1000	**0.027 (0.020)**	**0.012 (0.009)**	**0.001 (0.001)**
3x3 mixed	500	500	0.020 (0.022)	0.016 (0.019)	0.011 (0.024)

Table 2: Average estimation errors of γ, δ, θ for the 2x2 easy, 2x2 hard, and 3x3 mixed experiments. Values are reported as the mean over 500 Monte Carlo replicates, with standard deviations in parentheses. The best performing model in each experiment is indicated in bold.

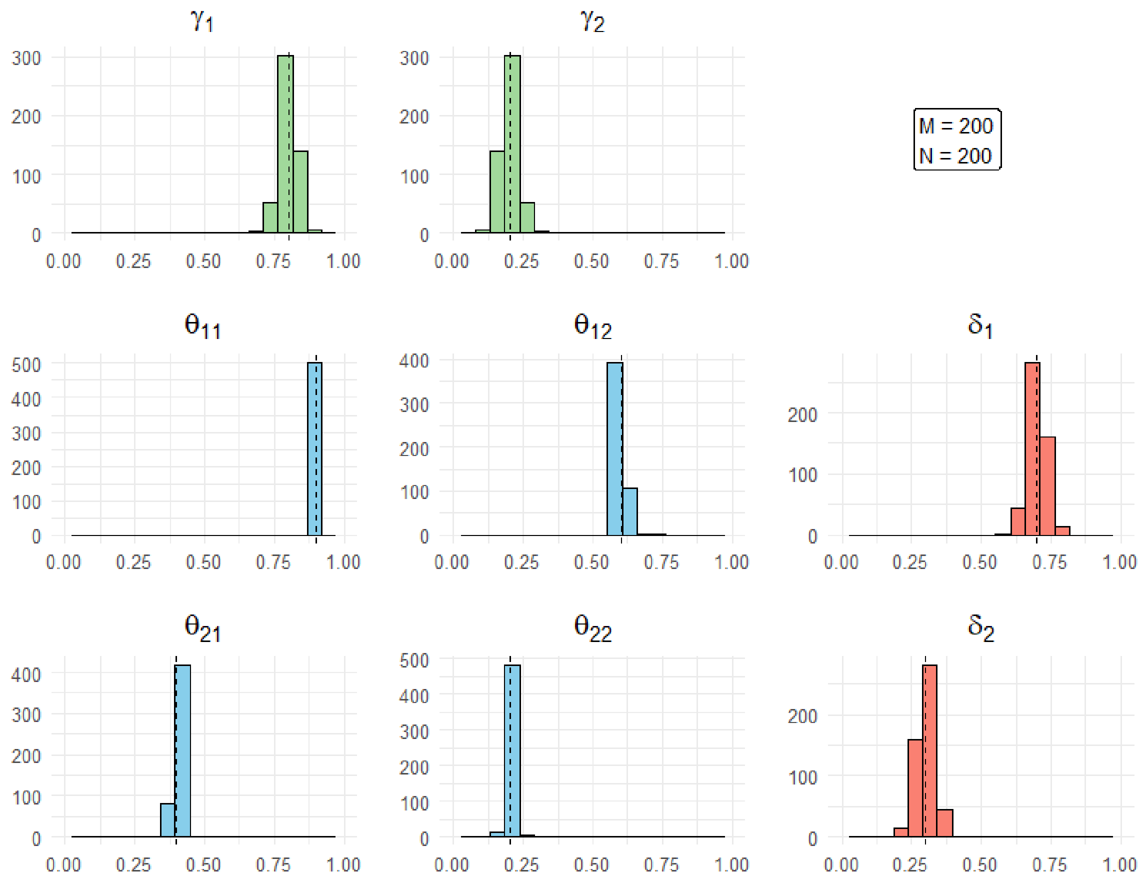

Figure 3: Histogram of estimated parameters (γ, δ, and θ) from 500 Monte Carlo replicates of the **2×2 easy** experiment with $N = 200$, $M = 200$. The dotted line indicates the true value of the respective parameter.

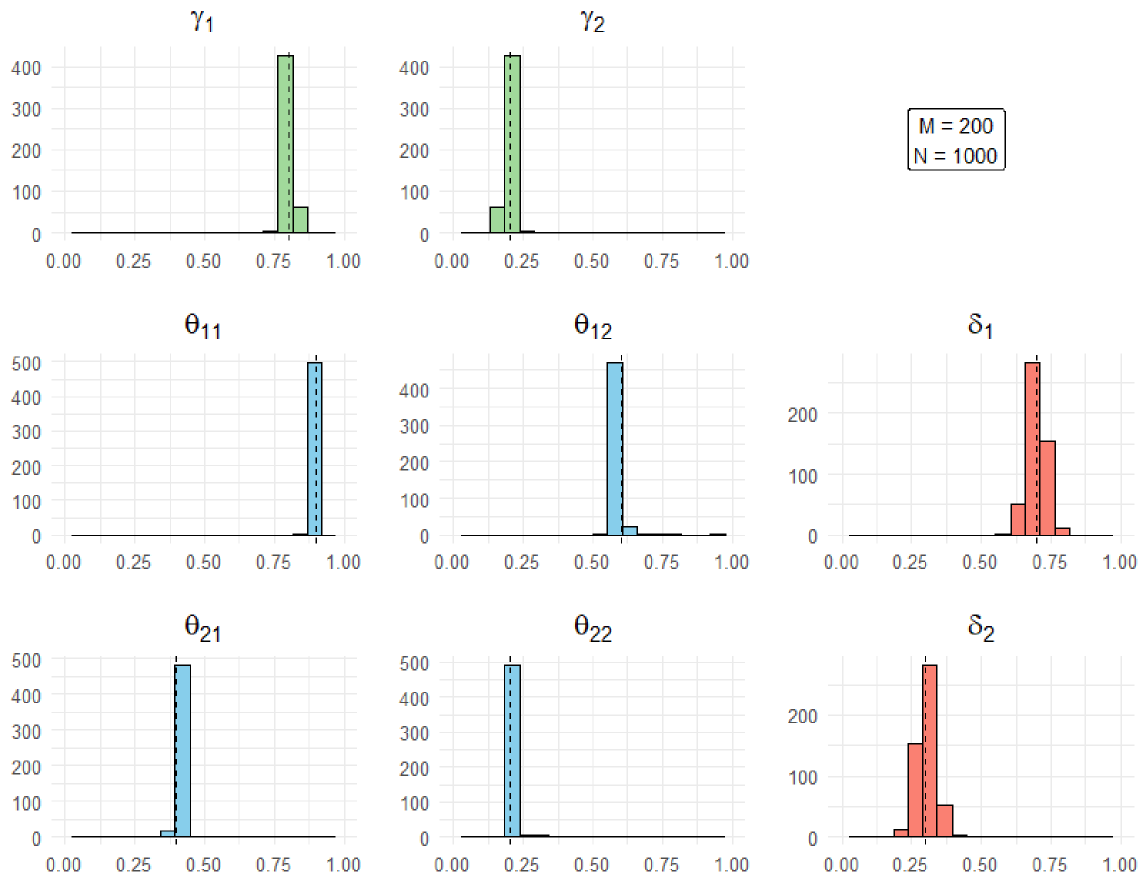

Figure 4: Histogram of estimated parameters (γ, δ, and θ) from 500 Monte Carlo replicates of the **2×2 easy** experiment with $N = 1000$, $M = 200$. The dotted line indicates the true value of the respective parameter.

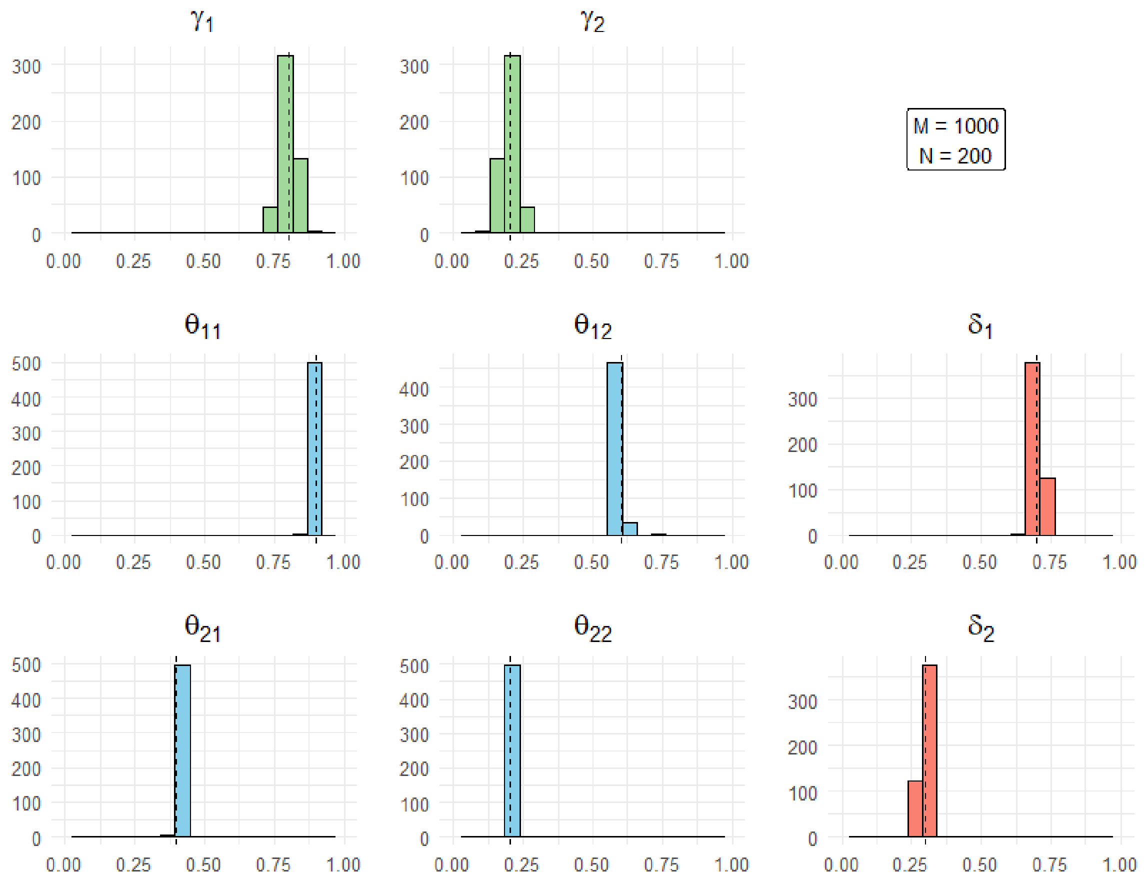

Figure 5: Histogram of estimated parameters (γ, δ, and θ) from 500 Monte Carlo replicates of the **2×2 easy** experiment with $N = 200$, $M = 1000$.

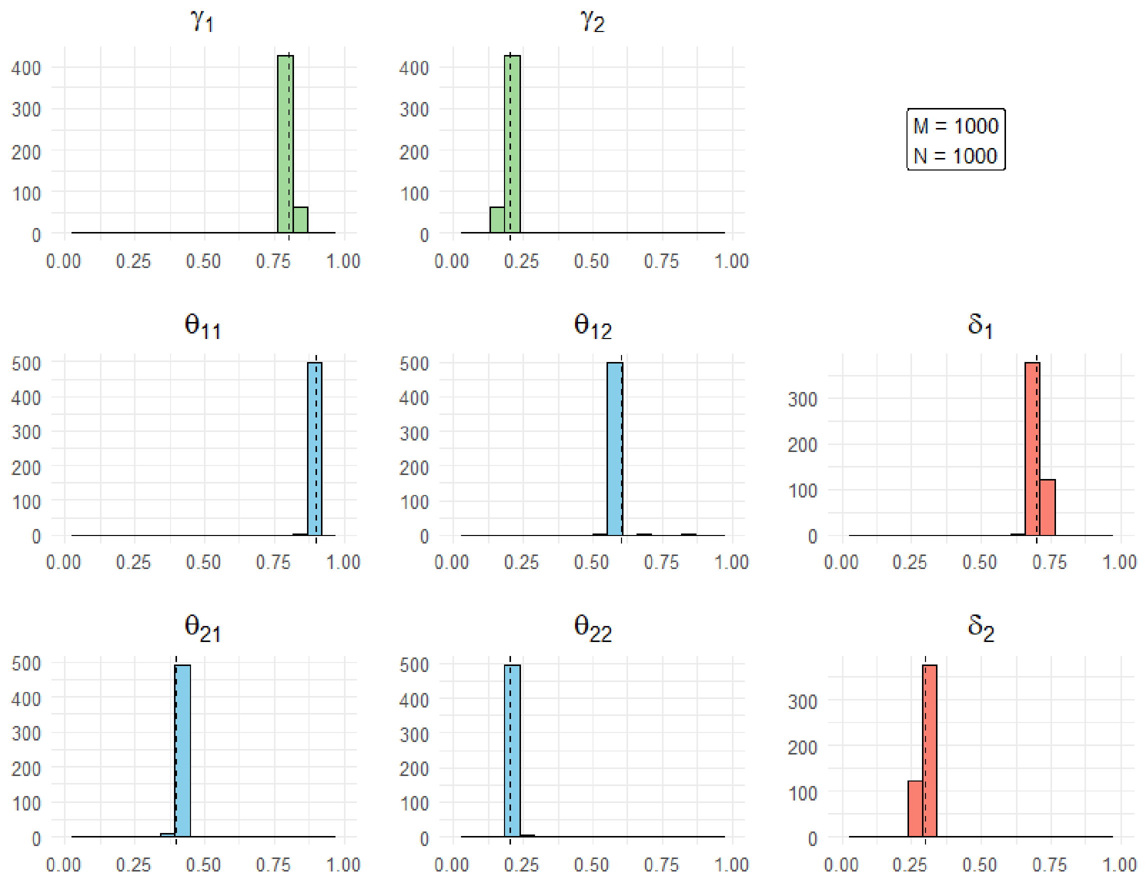

Figure 6: Histogram of estimated parameters (γ, δ, and θ) from 500 Monte Carlo replicates of the **2×2 easy** experiment with $N = 1000$, $M = 1000$. The dotted line indicates the true value of the respective parameter.

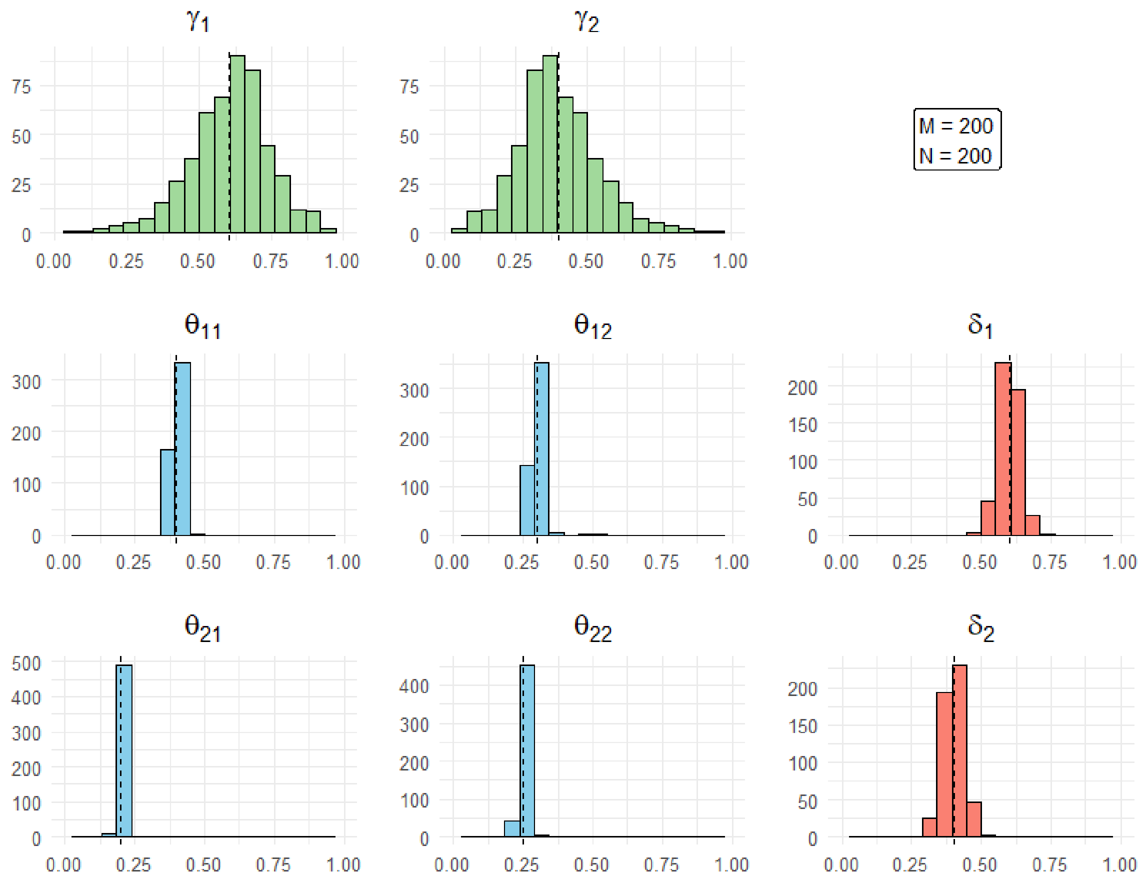

Figure 7: Histogram of estimated parameters (γ, δ, and θ) from 500 Monte Carlo replicates of the **2×2 hard** experiment with $N = 200$, $M = 200$. The dotted line indicates the true value of the respective parameter.

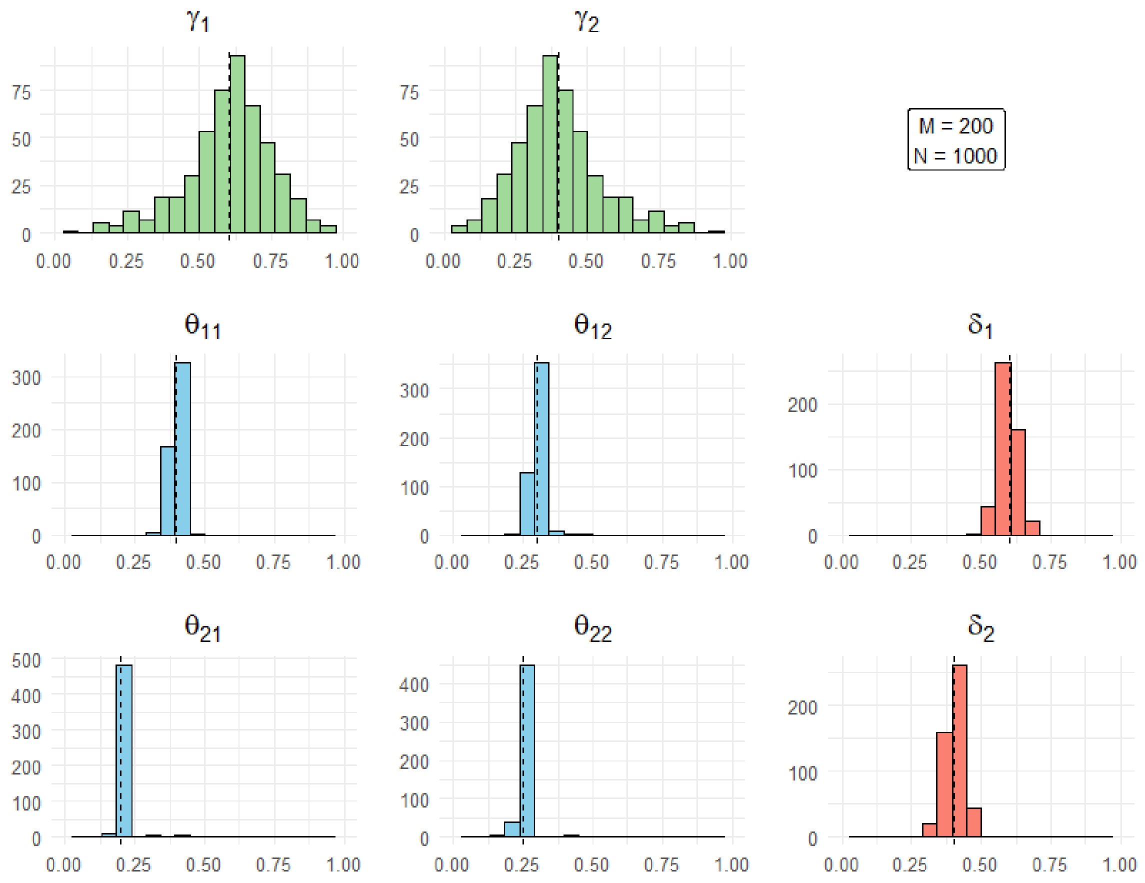

Figure 8: Histogram of estimated parameters (γ, δ, and θ) from 500 Monte Carlo replicates of the **2×2 hard** experiment with $N = 1000$, $M = 200$. The dotted line indicates the true value of the respective parameter.

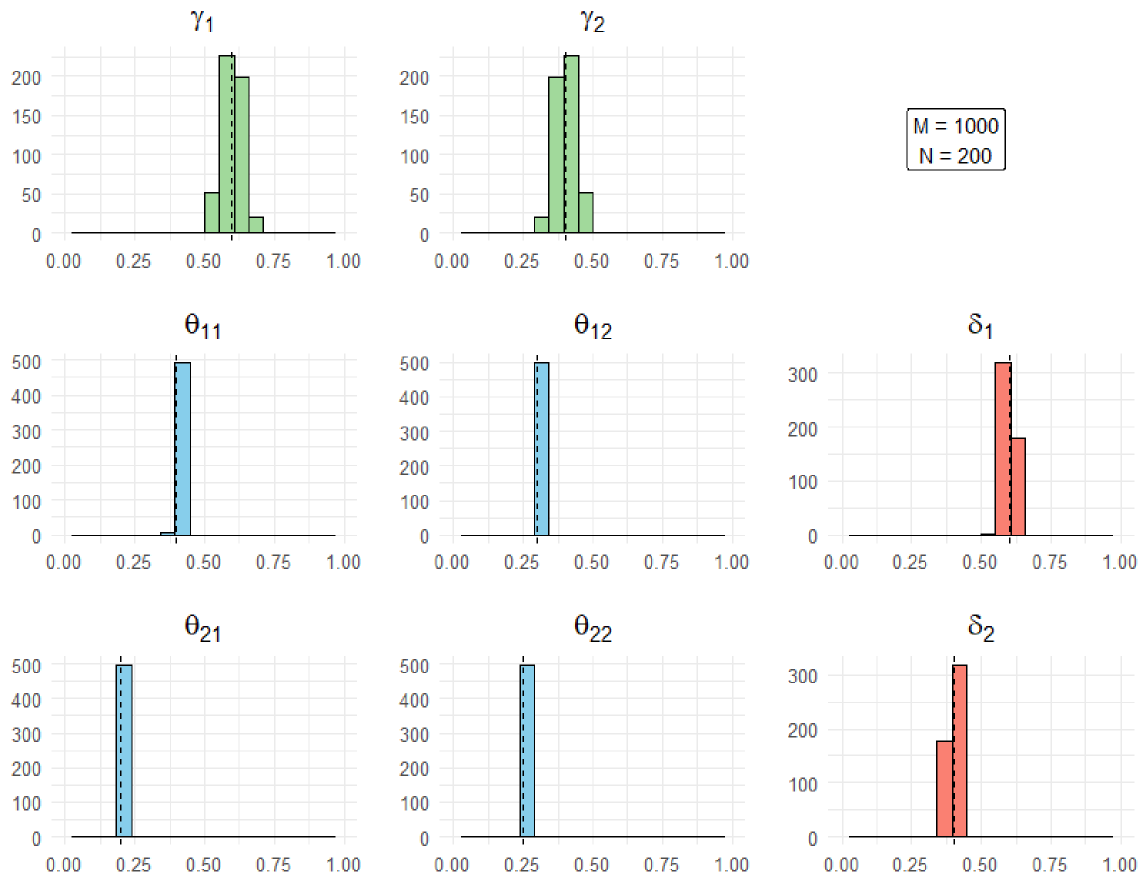

Figure 9: Histogram of estimated parameters (γ, δ, and θ) from 500 Monte Carlo replicates of the **2×2 hard** experiment with $N = 200$, $M = 1000$. The dotted line indicates the true value of the respective parameter.

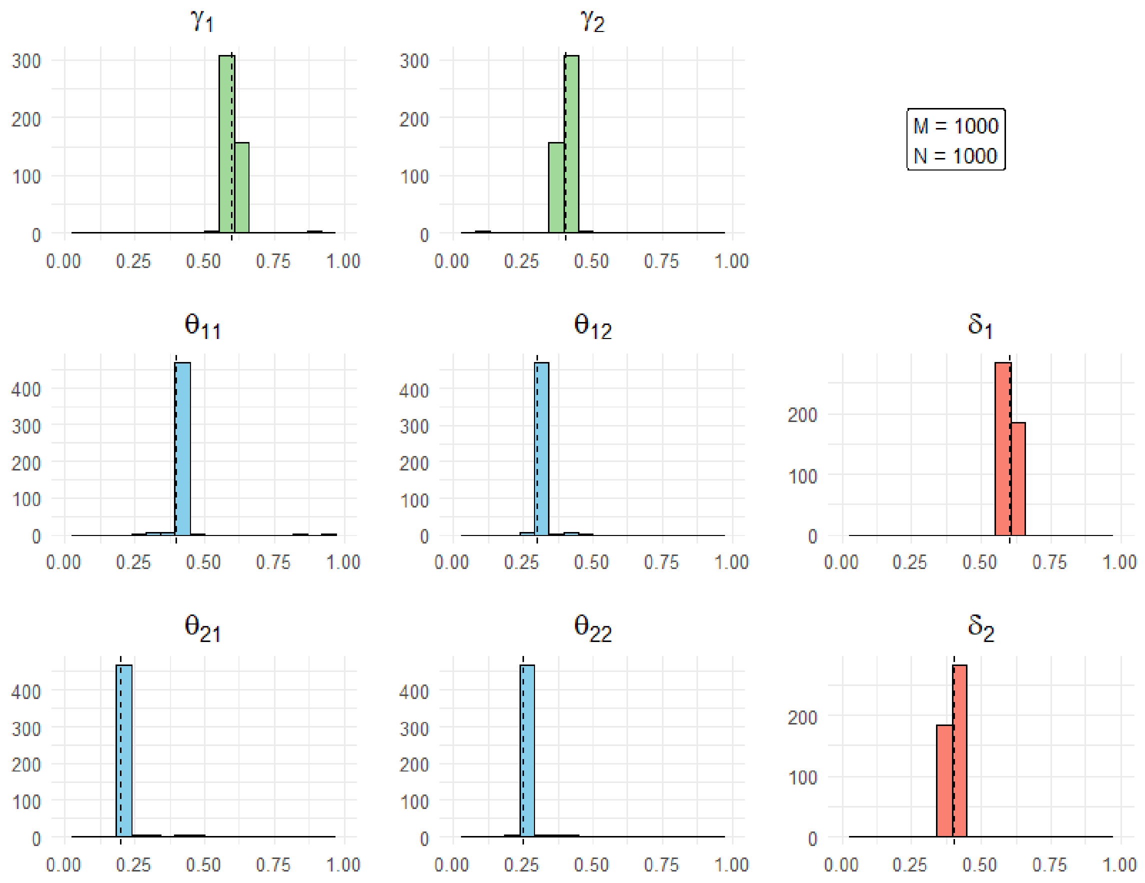

Figure 10: Histogram of estimated parameters (γ, δ, and θ) from 500 Monte Carlo replicates of the **2×2 hard** experiment with $N = 1000$, $M = 1000$. The dotted line indicates the true value of the respective parameter.

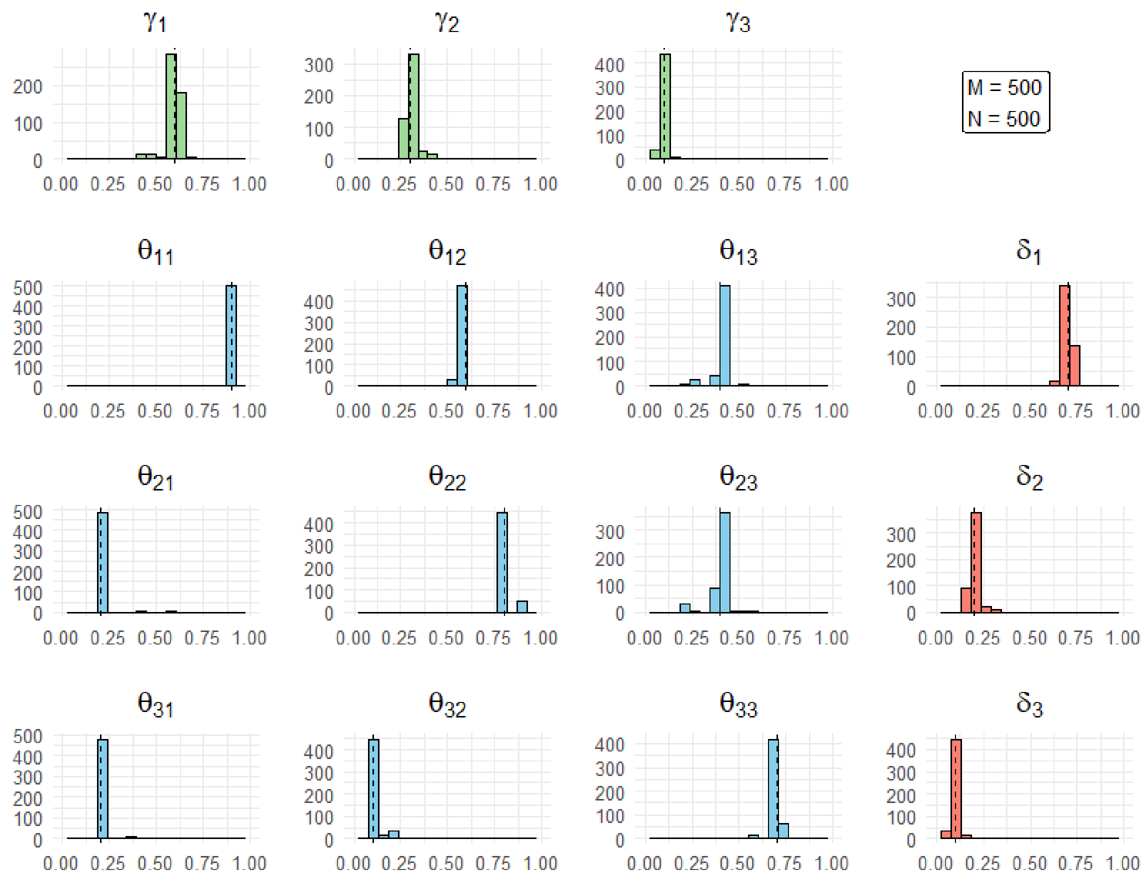

Figure 11: Histogram of estimated parameters (γ, δ, and θ) from 500 Monte Carlo replicates of the **3x3 mixed** experiment with $N = 500$, $M = 500$. The dotted line indicates the true value of the respective parameter.

In the 2x2 easy experiment (Figures 3 to 6), the average ℓ_1 error remained consistently low (below 0.025) across all sample sizes and parameters. As the number of nodes (N) increased, estimation of γ improved, whereas an increase of the hyperedges (M) led to a more accurate recovery of δ. Estimation of θ also improved with each increase in sample size (in any direction), although its error was generally low from the start. As seen in the monte carlo histograms (Figures 3 to 6), the estimates were generally centered around the true value for all parameters, with only rare and small deviations.

In the 2x2 hard experiment (Figures 7 to 10), estimation errors for γ were substantially higher when M was low, which is consistent with the drop of node clustering accuracy. Increasing M caused the error of γ to decrease, reaching levels comparable to the 2x2 easy case. The errors of δ, θ followed similar but less pronounced trends. The histograms (Figures 7 to 10) further illustrate this behavior, showing more dispersed estimates when M is low. This spread is particularly noticeable for γ in the $M = 200$ case. When $M = 1000$, the estimates for all parameters become tightly concentrated around their true values.

Finally, the 3x3 mixed experiment (Figure 11) resulted in slightly higher estimation errors across all parameters, which was expected from the added model complexity. However, this error increase was minor, suggesting that the BEM scales well in increased latent group dimensions. The histograms (Figure 11) support this observation, showing only slight deviations around the true parameter values.

3.5 Simulation 4 (model-selection)

In Simulation 4, we generated a hypergraph using a latent structure with $K = 3$ and $G = 3$. We then fitted models with varying numbers of latent clusters and computed the Integrated Complete-Data Likelihood (ICL) for each fitted model. The objective was to evaluate the ICL as a model selection criterion, that is, to identify whether the model with the highest ICL corresponds to the true latent structure.

The ICL criterion proved to be very effective selecting the correct number of latent group dimensions. Table 3 shows the relative frequency of each combination of K, G that maximized the ICL. On 93% of replicates, the ICL was maximized by a model matching the true latent structure. In the remaining cases, the selected model had a latent structure which differed from the true one only in one dimension, with the other dimension correctly specified. In addition, all misclassifications involved choosing a number of groups which was larger than the real one. In many of those cases, the additional group had a prior probability very close to zero, effectively resulting in a model which, in practice, has the correct number of groups.

K \ G	2	3	4
2	0	0	0
3	0	**0.93**	0.04
4	0	0.03	0

Table 3: Relative frequency of the latent structure that maximized the ICL over 500 Monte Carlo replicates.

3.6 The case for multiple random initializations

The BEM, as an EM variant, is prone to getting trapped in local maxima. To account for this, we used three random initializations for each replicate, and retained only the run that achieved the highest ICL value. This increases the chance of selecting a starting point which is not surrounded by poor local maxima.
Relying on a single initialization is risky and ill-advised. To illustrate the importance of multiple runs, we also report results for the three simulation scenarios (2x2 easy, 2x2 hard, and 3x3 mixed) using only the first BEM run in each replicate.

Dataset	**N**	**M**	Acc. Nodes	Acc. Hyperedges	ARI Nodes	ARI Hyperedges
2x2 easy	200	200	0.940 (0.146)	0.997 (0.034)	0.821 (0.352)	0.994 (0.077)
2x2 easy	200	1000	0.900 (0.226)	0.956 (0.205)	0.695 (0.528)	0.989 (0.099)
2x2 easy	1000	200	0.916 (0.170)	0.974 (0.117)	0.741 (0.402)	0.942 (0.233)
2x2 easy	1000	1000	0.937 (0.172)	0.982 (0.105)	0.789 (0.455)	0.995 (0.049)
2x2 hard	200	200	0.764 (0.131)	0.957 (0.108)	0.336 (0.248)	0.881 (0.217)
2x2 hard	200	1000	0.944 (0.078)	0.942 (0.090)	0.435 (0.646)	0.696 (0.460)
2x2 hard	1000	200	0.758 (0.139)	0.948 (0.151)	0.266 (0.299)	0.760 (0.427)
2x2 hard	1000	1000	0.951 (0.146)	0.971 (0.162)	0.721 (0.727)	0.916 (0.194)
3x3 mixed	500	500	0.944 (0.149)	0.908 (0.189)	0.947 (0.119)	0.918 (0.153)

Table 4: Classification accuracy and Adjusted Rand Index (ARI) for the 2x2 easy, 2x2 hard, and 3x3 mixed experiments, using only the first BEM run. Values are reported as the mean over 500 Monte Carlo replicates, with standard deviations in parentheses.

In Tables 4 and 5, we observe how using a single BEM initialization resulted in performance drops ranging from mild to severe. This decline was most evident in both node classification accuracy and parameter estimation error for the 2x2 hard experiment. Even in scenarios where mean performance was only slightly affected, the standard deviations were consistently much higher. This suggests that relying on a single run leads to unstable and less reliable estimates.

Dataset	**N**	**M**	$\ell_1(\boldsymbol{\gamma})$	$\ell_1(\boldsymbol{\delta})$	$\ell_1(\boldsymbol{\theta})$
2x2 easy	200	200	0.067 (0.13)	0.026 (0.25)	0.006 (0.005)
2x2 easy	200	1000	0.103 (0.222)	0.029 (0.104)	0.035 (0.078)
2x2 easy	1000	200	0.078 (0.15)	0.049 (0.110)	0.040 (0.080)
2x2 easy	1000	1000	0.075 (0.164)	0.016 (0.049)	0.021 (0.054)
2x2 hard	200	200	0.195 (0.139)	0.055 (0.104)	0.024 (0.032)
2x2 hard	200	1000	0.062 (0.143)	0.068 (0.158)	0.015 (0.044)
2x2 hard	1000	200	0.198 (0.143)	0.076 (0.143)	0.028 (0.040)
2x2 hard	1000	1000	0.040 (0.077)	0.032 (0.088)	0.007 (0.029)
3x3 mixed	500	500	0.129 (0.142)	0.165 (0.166)	0.122 (0.090)

Table 5: Average estimation errors of γ, δ, θ for the 2x2 easy, 2x2 hard, and 3x3 mixed experiments, using only the first BEM run. Values are reported as the mean over 500 Monte Carlo replicates, with standard deviations in parentheses.

4 Discussion

In this work, we derived a model-based approach for the co-clustering of binary hypergraphs with two latent structures, referred to as the Cocluster-LCA model. This model generalizes previous approaches that focused solely on clustering either the nodes or the hyperedges. In the Cocluster-LCA model, each node and hyperedge is assigned to a latent group, and the probability that a node participates in a hyperedge depends solely on their respective group memberships. This structure offers greater modeling flexibility and increases the interpretability of the resulting parameter estimates.

Such model-based co-clustering approaches have been studied in the literature (Govaert and Nadif, 2008). Our contribution lies in providing an extensive set of Monte Carlo simulations to compare the estimator's behavior under a wide range of settings, as well as a detailed, step-by-step analytical derivation of the re-estimation formulas, offering reproducibility for future applications.

The Cocluster-LCA takes the form of a finite mixture model, enabling the use of the Expectation Maximization for parameter estimation. However, because of the intractability of the E-step, a variational approximation (VEM) is instead used. A generalization of the VEM for co-clustering is implemented, referred to as the BEM. To further reduce the computational complexity of the variational E-step, a fixed-point update scheme is introduced, resulting in a more efficient estimation procedure. A number of Monte Carlo experiments were conducted to assess the estimator's ability to recover the true clustering structure and underlying parameter values.

The estimator's performance was strongly influenced by the sample sizes of both the nodes (N) and hyperedges (M). Specifically, increasing M improved the estimation of $\boldsymbol{\delta}$, while increasing N enhanced the estimation of $\boldsymbol{\gamma}$. As $\boldsymbol{\theta}$ depends on both nodes and hyperedges, its estimation improved with increases in either sample. Clustering accuracy for nodes and hyperedges followed a similar trend. Another factor affecting performance was the separability of the parameter values of the groups. Easily identifiable parameter configurations resulted in highly accurate estimates regardless of sample size, whereas more difficult to separate configurations ended up in poor estimation for smaller samples. Additionally, increasing the number of latent groups had a slight negative impact on both accuracy and parameter estimation. Lastly, the Integrated Complete Likelihood (ICL) criterion demonstrated strong performance in selecting the correct model structure.

There are several directions in which this work could be extended. First, the model could be augmented by allowing the node-joining probability to depend not only on the group memberships of the node and the hyperedge, but also on the groups of the other nodes participating in the hyperedge. This extension would enable the modeling of interactions where participation depends on both the nature of the event and the composition of its participants. Second, a temporal extension of the model could be considered, in which hyperedges are observed over time and the underlying parameters are either static or allowed to vary dynamically. In such a case, tools from survival analysis could be employed to model the timing of the hyperedge formation. This would allow the model to be applied to timestamped data, adding an extra layer of information to help understand how participation and group structure evolve over time.

A Optimization proofs for the VE and M steps of the BEM

The Φ criterion we aim to optimize in both VE and M steps of the BEM algorithm is

$$\Phi(\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\theta}) = \mathbb{E}_{Z,W|X}\left[L_C(\boldsymbol{\theta}, \boldsymbol{\delta}, \boldsymbol{\gamma}|X, Z, W)\right] + H(\boldsymbol{c}) + H(\boldsymbol{d})$$

where

$$\begin{aligned} H(\boldsymbol{c}) &= -\sum_{i=1}^{N}\sum_{k=1}^{K} c_i(k)log(c_i(k)) \\ H(\boldsymbol{d}) &= -\sum_{j=1}^{M}\sum_{g=1}^{G} d_j(g)log(d_j(g)) \end{aligned}$$

Expanding the terms we get

$$\begin{aligned} \Phi(\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{\gamma}, \delta, \theta) =& \sum_{j=1}^{M}\sum_{g=1}^{G} d_j(g)log(\delta_g) + \sum_{i=1}^{N}\sum_{k=1}^{K} c_i(k)log(\gamma_k) \\ &+ \sum_{j=1}^{M}\sum_{g=1}^{G}\sum_{i=1}^{N}\sum_{k=1}^{K} d_j(g)c_i(k)log(\theta_{gk}^{x_{ji}}(1-\theta_{gk})^{1-x_{ji}}) \\ &- \sum_{i=1}^{N}\sum_{k=1}^{K} c_i(k)log(c_i(k)) - \sum_{j=1}^{M}\sum_{g=1}^{G} d_j(g)log(d_j(g)) \end{aligned}$$

A.1 VE-step

A.1.1 Optimizing w.r.t. $c_i(k)$

We assume that $\boldsymbol{d}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\theta}$, are constant. As we have constrained optimization, we use the method of Lagrangian multipliers. Specifically, we have the N constraints:

$$g_i(\boldsymbol{c}) = 1 - \sum_{k=1}^{K} c_i(k) = 0 \qquad (29)$$

$$\forall i = 1, \ldots, N$$

The Lagrangian function, $\Lambda(\boldsymbol{c}, \lambda)$ them becomes the sum of the $N+1$ terms:

$$\Lambda(\boldsymbol{c}, \lambda) = \Phi(\boldsymbol{c}|\boldsymbol{d}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\theta}) + \sum_{i=1}^{N} g_i(c)\lambda_i$$

We take the derivative w.r.t. $c_i(k)$:

$$
\begin{aligned}
\frac{\partial \Lambda(\boldsymbol{c}, \lambda)}{\partial c_i(k)} &= \frac{\partial \Phi(\boldsymbol{c}|\boldsymbol{d}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\theta})}{\partial c_i(k)} + \frac{\partial \left(\sum\limits_{i=1}^{N} g_i(c_i)\lambda_i\right)}{\partial c_i(k)} \\
&= \frac{\partial \sum\limits_{j=1}^{M}\sum\limits_{g=1}^{G} d_j(g) log(\delta_g)}{\partial c_i(k)} + \frac{\partial \sum\limits_{i=1}^{N}\sum\limits_{k=1}^{K} c_i(k) log(\gamma_k)}{\partial c_i(k)} \\
&+ \frac{\partial \sum\limits_{j=1}^{M}\sum\limits_{g=1}^{G}\sum\limits_{i=1}^{N}\sum\limits_{k=1}^{K} d_j(g) c_i(k) log(\theta_{gk}^{x_{ji}}(1-\theta_{gk})^{1-x_{ji}})}{\partial c_i(k)} \\
&- \frac{\partial \sum\limits_{i=1}^{N}\sum\limits_{k=1}^{K} c_i(k) log(c_i(k))}{\partial c_i(k)} - \frac{\partial \sum\limits_{j=1}^{M}\sum\limits_{g=1}^{G} d_j(g) log(d_j(g))}{\partial c_i(k)} + \sum_{i=1}^{N}\lambda_i \sum_{k=1}^{K}(1 - c_i(k)) \\
&= 0 + log(\gamma_k) + \sum_{j=1}^{M}\sum_{g=1}^{G} d_j(g) log(\theta_{gk}^{x_{ji}}(1-\theta_{gk})^{1-x_{ji}}) - (1 + log(c_i(k))) - \lambda_i \\
&= log(\gamma_k) + log\left(\prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{x_{ji}}(1-\theta_{gk})^{1-x_{ji}}\right)^{d_j(g)} - 1 - log(c_i(k)) - \lambda_i \\
&= log(\gamma_k) + log\left(\prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right) - 1 - log(c_i(k)) - \lambda_i \\
&= log\left(\gamma_k \prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right) - 1 - log(c_i(k)) - \lambda_i
\end{aligned}
$$

Setting the obtained quantity equal to 0 yields:

$$
c_{ik} = e^{log\left(\gamma_k \prod\limits_{j=1}^{M}\prod\limits_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right) - 1 - \lambda_i}
$$

Returning to the constraint equation, (29),

$$\begin{aligned}
1 - \sum_{k=1}^{K} c_i(k) &= 0 \\
1 - \sum_{k=1}^{K} e^{log\left(\gamma_k \prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right)-1-\lambda_i} &= 0 \\
e^{-\lambda_i - 1}\sum_{k=1}^{K} e^{log\left(\gamma_k \prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right)} &= 1 \\
e^{-\lambda_i - 1}\sum_{k=1}^{K} \left(\gamma_k \prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right) &= 1 \\
e^{-\lambda_1 - 1} = \frac{1}{\sum\limits_{k=1}^{K} \gamma_k \prod\limits_{j=1}^{M}\prod\limits_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}} \\
\lambda_i = -log\left(\frac{1}{\sum\limits_{k=1}^{K} \gamma_k \prod\limits_{j=1}^{M}\prod\limits_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}}\right) - 1 \\
\lambda_i = log\left(\sum_{k=1}^{K} \gamma_k \prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right) - 1
\end{aligned}$$

Plugging the λ_i back to the derived solution we obtain:

$$\begin{aligned}
\hat{c}_i(k) &= e^{log\left(\gamma_k \prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right)-1-\lambda_i} \\
&= e^{log\left(\gamma_k \prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right)-\left[log\left(\sum_{k=1}^{K}\gamma_k \prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right)-1\right]-1} \\
&= \frac{e^{log\left(\gamma_k \prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right)}}{e^{log\left(\sum_{k=1}^{K}\gamma_k \prod_{j=1}^{M}\prod_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}\right)}} \\
&= \frac{\gamma_k \prod\limits_{j=1}^{M}\prod\limits_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}}{\sum\limits_{i=1}^{K}\gamma_k \prod\limits_{j=1}^{M}\prod\limits_{g=1}^{G} \theta_{gk}^{d_j(g)x_{ji}}(1-\theta_{gk})^{d_j(g)(1-x_{ji})}}
\end{aligned}$$

A.1.2 Optimizing w.r.t $d_j(g)$

The proof is equivalent to the one for $c_i(k)$.

We assume that $\boldsymbol{c}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\theta}$ are constant. As we have constrained optimization, we use the method of Lagrangian multipliers. Specifically, we have the M constraints:

$$h_j(\boldsymbol{d}) = 1 - \sum_{g=1}^{G} d_j(g) = 0 \qquad (30)$$
$$\forall j = 1, \ldots, M$$

The Lagrangian function, $\Lambda(d, \lambda)$ then becomes the sum of the $M + 1$ terms:

$$\Lambda(\boldsymbol{d}, \lambda) = \Phi(\boldsymbol{d}|\boldsymbol{c}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\theta}) + \sum_{j=1}^{M} h_j(d)\lambda_j$$

We take the derivative w.r.t. $d_j(g)$:

$$\begin{aligned}
\frac{\partial\Lambda(\boldsymbol{d}, \lambda)}{\partial d_j(g)} &= \frac{\partial\Phi(\boldsymbol{d}|\boldsymbol{c}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\theta})}{\partial d_j(g)} + \frac{\partial\left(\sum\limits_{j=1}^{M} h_j(\boldsymbol{d})\lambda_j\right)}{\partial d_j(g)} \\
&= \frac{\partial \sum\limits_{i=1}^{N}\sum\limits_{k=1}^{K} c_i(k)\log(\gamma_k)}{\partial d_j(g)} + \frac{\partial \sum\limits_{g=1}^{G} d_j(g)\log(\delta_g)}{\partial d_j} \\
&\quad + \frac{\partial \sum\limits_{i=1}^{N}\sum\limits_{k=1}^{K}\sum\limits_{j=1}^{M}\sum\limits_{g=1}^{G} d_j(g)c_i(k)\log(\theta_{gk}^{x_{ji}}(1-\theta_{gk})^{1-x_{ji}})}{\partial d_j} \\
&\quad - \frac{\partial \sum\limits_{g=1}^{G} d_j(g)\log(d_j(g))}{\partial d_j} - \frac{\partial \sum\limits_{i=1}^{N}\sum\limits_{k=1}^{K} c_i(k)\log(c_i(k))}{\partial d_j(g)} + \sum_{j=1}^{M}\lambda_j\sum_{g=1}^{G}(1 - d_j(g)) \\
&= 0 + \log(\delta_g) + \sum_{i=1}^{N}\sum_{k=1}^{K} c_i(k)\log(\theta_{gk}^{x_{ji}}(1-\theta_{gk})^{1-x_{ji}}) - (1 + \log(d_j(g))) - \lambda_j \\
&= \log(\delta_g) + \log\left(\prod_{i=1}^{N}\prod_{k=1}^{K}\theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right) - 1 - \log(d_j(g)) - \lambda_j \\
&= \log\left(\delta_g\prod_{i=1}^{N}\prod_{k=1}^{K}\theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right) - 1 - \log(d_j(g)) - \lambda_j
\end{aligned}$$

Setting equal to 0 yields:

$$d_{jg} = e^{\log\left(\delta_g\prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K}\theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right)-1-\lambda_j}$$

Returning to the constraint equation (30):

$$
\begin{aligned}
1 - \sum_{g=1}^{G} d_j(g) &= 0 \\
1 - \sum_{g=1}^{G} e^{\log\left(\delta_g \prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right)-1-\lambda_j} &= 0 \\
e^{-\lambda_j-1}\sum_{g=1}^{G} e^{\log\left(\delta_g \prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right)} &= 1 \\
e^{-\lambda_j-1}\sum_{g=1}^{G}\left(\delta_g \prod_{i=1}^{N}\prod_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right) &= 1 \\
e^{-\lambda_j-1} = \frac{1}{\sum\limits_{g=1}^{G}\delta_g \prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}} \\
\lambda_j = -\log\left(\frac{1}{\sum\limits_{g=1}^{G}\delta_g \prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}}\right) &- 1 \\
\lambda_j = \log\left(\sum_{g=1}^{G}\delta_g \prod_{i=1}^{N}\prod_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right) &- 1
\end{aligned}
$$

Plugging λ_j back into the derived solution we obtain:

$$
\begin{aligned}
d_{jg} &= e^{\log\left(\delta_g \prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right)-1-\lambda_j} \\
&= e^{\log\left(\delta_g \prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right)-\left[\log\left(\sum\limits_{g=1}^{G}\delta_g \prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right)-1\right]-1} \\
&= \frac{e^{\log\left(\delta_g \prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right)}}{e^{\log\left(\sum\limits_{g=1}^{G}\delta_g \prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}\right)}} \\
&= \frac{\delta_g \prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}}{\sum\limits_{g=1}^{G}\delta_g \prod\limits_{i=1}^{N}\prod\limits_{k=1}^{K} \theta_{gk}^{c_i(k)x_{ji}}(1-\theta_{gk})^{c_i(k)(1-x_{ji})}}
\end{aligned}
$$

A.2 M-step

A.2.1 Optimizing w.r.t. γ_k

This is a constrained optimization problem with the single constraint: $h(\boldsymbol{\gamma}) := 1 - \sum_{k=1}^{K} \gamma_k = 0$. Define the Langrangian $\Lambda(\boldsymbol{\gamma}, \lambda) = \Phi(\boldsymbol{\gamma}|\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{\delta}, \boldsymbol{\theta}) + h(\boldsymbol{\gamma})\lambda$

$$\begin{aligned}
\frac{\partial \Lambda(\boldsymbol{\gamma}, \lambda)}{\partial \gamma_k} &= \frac{\partial \Phi(\boldsymbol{\gamma}|\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{\delta}, \boldsymbol{\theta})}{\partial \gamma_k} + \frac{\partial \left(h(\boldsymbol{\gamma})\lambda\right)}{\partial \gamma_k} \\
&= \frac{\partial \left(\sum_{i=1}^{N} \sum_{k=1}^{K} c_i(k) log(\gamma_k) \right)}{\partial \gamma_k} + \lambda \frac{\partial \left(1 - \sum_{k=1}^{K} \gamma_k \right)}{\partial \gamma_k} \\
&= \frac{\sum_{i=1}^{N} c_i(k)}{\gamma_k} - \lambda
\end{aligned}$$

Setting it equal to 0, solving for γ_k and replacing it in the constraint yields:

$$\gamma_k = \frac{\sum_{i=1}^{N} c_i(k)}{\lambda} \Leftrightarrow 1 - \frac{\sum_{k=1}^{K} \sum_{i=1}^{N} c_i(k)}{\lambda} = 0 \Leftrightarrow \lambda = \sum_{k=1}^{K} \sum_{i=1}^{N} c_i(k)$$

Note that $\sum_{k=1}^{K} c_i(k) = 1$ for all i, and thus $\lambda = N$, which we replace in the original solution to get

$$\hat{\gamma}_k = \frac{\sum_{i=1}^{N} c_i(k)}{N}$$

A.2.2 Optimizing w.r.t. δ_g

A constrained optimization problem with constraint $h(\delta) = 1 - \sum_{g=1}^{G} \delta_g = 0$. Define the Langrangian $\Lambda(\boldsymbol{\delta}, \lambda) = \Phi(\boldsymbol{\delta}|\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{\gamma}, \boldsymbol{\theta}) + h(\boldsymbol{\delta})\lambda$

$$\begin{aligned}
\frac{\partial \Lambda(\boldsymbol{\delta}, \lambda)}{\partial \delta_g} &= \frac{\partial \Phi(\boldsymbol{\delta}|\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{\gamma}, \boldsymbol{\theta})}{\partial \delta_g} + \frac{\partial \left(h(\boldsymbol{\delta})\lambda\right)}{\partial \delta_g} \\
&= \frac{\partial \left(\sum_{j=1}^{M} \sum_{g=1}^{G} d_j(g) log \delta_g) \right)}{\partial \delta_g} + \lambda \frac{\partial \left(1 - \sum_{g=1}^{G} \delta_g \right)}{\partial \delta_g} \\
&= \frac{\sum_{j=1}^{M} d_j(g)}{\delta_g} - \lambda
\end{aligned}$$

Setting it equal to 0, solving for δ_g and replacing it in the constraint yields:

$$\delta_g = \frac{\sum\limits_{j=1}^{M} d_j(g)}{\lambda} \Leftrightarrow 1 - \frac{\sum\limits_{g=1}^{G}\sum\limits_{j=1}^{M} d_j(g)}{\lambda} = 0 \Leftrightarrow \lambda = \sum_{g=1}^{G}\sum_{j=1}^{M} d_j(g)$$

Note that $\sum\limits_{g=1}^{G} d_j(g) = 1$ for all i, and thus $\lambda = M$, which we replace in the original solution to get

$$\hat{\delta}_g = \frac{\sum\limits_{j=1}^{M} d_j(g)}{M}$$

A.2.3 Optimizing w.r.t. θ_{gk}

This problem is an unconstrained optimization problem. Thus we solve it by taking the partial derivative w.r.t. θ_{gk} and setting it equal to 0.

$$\begin{aligned}
\frac{\partial \Phi(\boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\theta} | \boldsymbol{c}, \boldsymbol{d})}{\partial \theta_{gk}} &= \sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)\frac{\partial\left[log(\theta_{gk}^{x_{ji}}(1-\theta_{gk})^{1-x_{ji}})\right]}{\partial \theta_{gk}} \\
&= \sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)\left[\partial\frac{x_{ji}log(\theta_{gk})}{\partial \theta_{gk}} + \partial\frac{(1-x_{ji})log(1-\theta_{gk})}{\partial \theta_{gk}}\right] \\
&= \sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)\left[\frac{x_{ji}}{\theta_{gk}} - \frac{(1-x_{ji})}{1-\theta_{gk}}\right] \\
&= \sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)\left[\frac{x_{ji}-\theta_{gk}}{\theta_{gk}(1-\theta_{gk})}\right] \\
&= \sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)\frac{x_{ji}}{\theta_{gk}(1-\theta_{gk})} - \sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)\frac{\theta_{gk}}{\theta_{gk}(1-\theta_{gk})} \\
&= \frac{1}{\theta_{gk}(1-\theta_{gk})}\left[\sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)x_{ji} - \theta_{gk}\sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)\right]
\end{aligned}$$

And by setting the obtained quantity equal to 0 we obtain:

$$\begin{aligned}
\frac{1}{\hat{\theta}_{gk}(1-\hat{\theta}_{gk})}\left[\sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)x_{ji} - \hat{\theta}_{gk}\sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)\right] &= 0 \\
\Rightarrow \sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)x_{ji} - \hat{\theta}_{gk}\sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k) &= 0 \\
\Rightarrow \hat{\theta}_{gk}\sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k) &= \sum_{i=1}^{N}\sum_{j=1}^{M} d_j(g)c_i(k)x_{ji} \\
\Rightarrow \hat{\theta}_{gk} &= \frac{\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{M} d_j(g)c_i(k)x_{ji}}{\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{M} d_j(g)c_i(k)}
\end{aligned}$$

References

Arora, N. and Ventresca, M. (2018). A network-based approach for modeling and analyzing supply chain systems. *Applied Network Science*, 3(1):1–20.

Battiston, F., Cencetti, G., Iacopini, I., Latora, V., Lucas, M., Patania, A., Young, J.-G., and Petri, G. (2020). Networks beyond pairwise interactions: Structure and dynamics. *Physics Reports*, 874:1–92.

Biernacki, C., Celeux, G., and Govaert, G. (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):719–725.

Biernacki, C., Celeux, G., and Govaert, G. (2003). Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate gaussian mixture models. *Computational Statistics Data Analysis*, 41(3):561–575. Recent Developments in Mixture Model.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Brault, V., Keribin, C., Celeux, G., and Govaert, G. (2014). Estimation and selection for the latent block model on categorical data. *Statistics and Computing*, 25:1–16.

Brusa, L. and Matias, C. (2024). Model-based clustering in simple hypergraphs through a stochastic blockmodel. *Scandinavian Journal of Statistics*, 51(4):1661–1684.

Celeux, G. and Govaert, G. (1985). A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics Quarterly*, 2(1):73–82.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

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

Gyllenberg, M., Koski, T., Reilink, E., and Verlaan, M. (1994). Non-uniqueness in probabilistic numerical identification of bacteria. *Journal of Applied Probability*, 31(2):542–548.

Keribin, C., Brault, V., Celeux, G., and Govaert, G. (2012). Model selection for the binary latent block model. In *20th International Conference on Computational Statistics (COMPSTAT 2012)*, pages 379–390, Limassol, Cyprus.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Neal, R. and Hinton, G. (2000). A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, 89.

Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, 45(2):167–256.

Ng, T. L. J. and Murphy, T. B. (2022). Model-based clustering for random hypergraphs. *Advances in Data Analysis and Classification*, 16(3):691–723.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Rubin, D. B. and Little, R. J. A. (1991). Statistical analysis with missing data. *Journal of Educational Statistics*, 16(2):150–155.

Shireman, E., Steinley, D., and Brusco, M. (2015). Examining the effect of initialization strategies on the performance of gaussian mixture modeling. *Behavior Research Methods*, 49.

Vinh, N. X., Epps, J., and Bailey, J. (2009). Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1073–1080, New York, NY, USA. Association for Computing Machinery.

Wu, C. F. J. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103.

Xiao, H.-B., Hu, F., Li, P.-Y., Song, Y.-R., and Zhang, Z.-K. (2024). Information propagation in hypergraph-based social networks. *Entropy*, 26(11).