JDSSV

Journal of Data Science, Statistics, and Visualisation

August 2024, Volume IV, Issue IV. *doi: 10.52933/jdssv.v4i4.106*

Deep Dynamic Co-Clustering of Count Data Streams: Application to Pharmacovigilance

Giulia Marchello
Inria

Alexandre Destere
Université Côte d'Azur Medical Centre

Marco Corneli
Université Côte d'Azur

Charles Bouveyron
Université Côte d'Azur

Abstract

Co-clustering is a widely used technique for analyzing complex and high-dimensional data across various domains. However, traditional models focus on continuous and dense data in fixed time frames, where cluster assignments remain static. Also, they often require all data to be in memory, posing issues for large datasets. The proposed online co-clustering model addresses this by processing data incrementally. We introduce a novel inference process for the latent block model to handle online co-clustering of sparse data matrices. This model assumes observations follow a time and block dependent mixture of zero-inflated distributions, combining stochastic processes with time-varying sparsity modeling. We use Bayesian online change point detection to detect abrupt changes in cluster memberships and data sparsity. The inference process employs a unique variational procedure, with the maximization step training an LSTM neural network to solve the dynamical systems. Numerical experiments on simulated datasets show the effectiveness of our methodology for count data streams. Applied to a large-scale dataset from the Regional Center of Pharmacovigilance of Nice (France), the model provides meaningful online segmentation of drugs and adverse drug reactions.

Keywords: Co-clustering, ZIP distribution, online inference, VEM algorithm, data streams.

1. Introduction

The need for unsupervised machine learning techniques, like clustering, is growing, particularly in pharmacovigilance, where detecting drug-associated adverse events is critical for ensuring drug safety. Current methods heavily rely on manual analysis, which can be incomplete due to the sheer volume of data. Automated clustering methods can effectively summarize data to detect safety signals over time, freeing pharmacovigilance experts to focus on critical tasks. Beyond pharmacovigilance, clustering methods are vital in social media, e-commerce, and biomedical data due to their ability to handle high-dimensional, sparse datasets. Co-clustering, which clusters both observations and features simultaneously, is especially useful for summarizing complex data structures. In dynamic environments like pharmacovigilance, where data patterns evolve over time due to new drugs or changing drug effects, dynamic co-clustering methods are essential. Online change point detection algorithms can identify shifts in data patterns, triggering timely investigations. This paper proposes an online model-based co-clustering approach for real-time safety signal detection from adverse drug reaction notifications. By analyzing count data over time, our method identifies temporal breaks in safety signals, facilitating alerts and further investigations by medical authorities. The aim is to demonstrate the method's potential as a routine tool in pharmacovigilance.

1.1. Related work

This section summarizes the related work in dynamic co-clustering and change point detection.

Co-clustering and Latent Block Models. Co-clustering is a versatile method for analyzing datasets by simultaneously clustering both observations and features. Various co-clustering approaches exist, categorized into metric-based methods like non-negative matrix tri-factorization (NMTF) (Labiod and Nadif 2011; Ding et al. 2006), spectral co-clustering (Dhillon 2001), information theory (Dhillon et al. 2003), and model-based approaches (e.g., Bouveyron et al. 2019). Model-based co-clustering, in particular, is valued for its robust statistical foundations and adaptability to diverse data types and levels of sparsity. The latent block model (LBM) (Govaert and Nadif 2003) is foundational in model-based co-clustering, initially designed for binary data matrices. LBM assumes rows and columns are grouped into hidden clusters, with observations within a block (intersection of a row cluster and a column cluster) being independent and identically distributed. Over the last two decades, LBM has been extended to accommodate count data (Govaert and Nadif 2010), continuous data (Lomet 2012), categorical data (Keribin et al. 2015), ordinal data (Jacques and Biernacki 2018; Corneli et al. 2020), functional data (Bouveyron et al. 2018), textual data (Bergé et al. 2019), and mixed-type data (Selosse et al. 2020). For a comprehensive overview, Biernacki et al. (2023) provide a detailed survey of LBM and its applications. Recently, Boutalbi et al. (2020) introduced the tensor latent block model (TLBM), which co-clusters rows and columns of a 3D matrix where covariates represent the third dimension. TLBM is implemented across various datasets including continuous (Gaussian TLBM), binary (Bernoulli TLBM), and contingency tables (Poisson TLBM), demonstrating its versatility and applicability.

Dynamic models for clustering and co-clustering. Dynamic clustering models have gained attention in recent years, particularly in extending static model-based techniques to handle temporal data. While static model-based clustering and co-clustering have been extensively studied, dynamic models represent a more recent development. Notably, more work has been done in dynamic network clustering, particularly with the Stochastic Block Model (SBM) (Nowicki and Snijders 2001), which is a special case of the latent block model (LBM) adapted for non-square and non-symmetric data matrices. Yang et al. (2011) introduced a dynamic version of SBM where nodes can switch clusters over time in a Markovian framework, using transition probabilities collected into a transition matrix. Matias and Miele (2017) demonstrated that in dynamic SBMs, varying both connectivity parameters and cluster memberships over time can lead to identifiability issues. Recently, Marchello et al. (2022) proposed extensions of LBM for dynamic co-clustering, including a model for three-dimensional counting tensors that clusters rows, columns, and slices simultaneously. Their subsequent work, Zip-dLBM, allows observations and features to switch clusters dynamically over time, using zero-inflated distributions to handle highly sparse datasets (Marchello et al. 2024). In a different approach, Casa et al. (2021) extended LBM to longitudinal data using a shape invariant model, while Boutalbi et al. (2021) developed a model-based co-clustering method for sparse three-way data, treating the third dimension as discrete and temporal. These methods primarily facilitate macroscopic analysis as they do not explicitly capture the temporal dependence of variables.

Change point detection. Change points indicate abrupt shifts in the pattern of a time series dataset, crucial for analyzing and forecasting such data. They pinpoint significant moments when the underlying process generating the data undergoes a noticeable change. Numerous change point detection algorithms exist in the literature, categorized into online and offline methods. Online algorithms operate in real-time as data streams in, making them suitable for detecting pharmacovigilance events promptly. Among online methods, likelihood and probabilistic approaches have proven effective (Kondratev et al. 2022; Kavitha and Punithavalli 2010). A seminal approach is Bayesian Online Change Point Detection (BOCD) introduced by Adams and MacKay (2007). BOCD assesses the likelihood of a run length increasing with each new data point and resets when a change point is detected. Another approach, by Kawahara and Sugiyama (2009), employs subspace identification within state-space models to detect change points based on time series data.

1.2. Our contribution

This paper presents an online extension of Zip-dLBM (Marchello et al. 2024), a co-clustering method tailored for evolving data matrices with sparsity. Three key innovations are introduced: first, an online estimation algorithm capable of processing data streams; second, integration of Bayesian Online Change Point Detection (BOCD) to identify abnormal events affecting data generation; third, adoption of LSTM networks in place of fully connected neural networks for improved modeling of time-evolving parameters. The code repository is accessible at `https://github.com/giuliamar95/Stream-Zip-dLBM`.

1.3. Organization of the paper

This paper is organized as follows. Section 2 recalls the generative model Zip-dLBM. Section 3 introduces the proposed online inference for stream data. Section 4 presents various experiments on simulated data to test and evaluate the model performances. In Section 5, an application on a real ADRs dataset is presented to illustrate the potential of Stream Zip-dLBM in pharmacovigilance. Section 6 provides some concluding remarks.

2. Stream Zip-dLBM

The following section outlines the zero-inflated Poisson Dynamic Latent Block Model (Zip-dLBM, Marchello et al. 2024) for batch processing. Although applicable to any zero-inflated distribution, we focus on the zero-inflated Poisson (ZIP) distribution due to its relevance in pharmacovigilance count data. In Zip-dLBM, data are collected into time-evolving matrices over the interval $[0, T]$, with equally spaced time points:

$$0 = t_0 < t_1 < t_u \leq t_U = T.$$

We denote each time point as t_u or simply t. At time t, the incidence matrix $X(t) \in \mathbb{N}^{N \times M}$ has $X_{ij}(t)$ as its element, representing the number of interactions between observation i and feature j from $t-1$ to t. Rows of $X(t)$ are indexed by $i = \{1, \ldots, N\}$ and columns by $j = \{1, \ldots, M\}$. The goal of Zip-dLBM is to cluster the rows and columns of the series of data matrices $\{X(t)\}_t$ for $t \in [0, T]$.

2.1. A zero-inflated dynamic latent block model

Consider a fixed time interval $[0, T]$ with a non-online model. The goal of Zip-dLBM is to cluster the rows and columns of time-varying data matrices $\{X(t)\}_t$. The numbers of clusters, Q for rows and L for columns, remain constant, but memberships can change over $[0, T]$.

Cluster modeling

We use evolving random matrices, $Z(t)$ for rows and $W(t)$ for columns, to track cluster memberships. $Z(t) \in \{0,1\}^{N \times Q}$, with $Z_i(t)$ denoting the i-th row, and $W(t) \in \{0,1\}^{M \times L}$, with $W_j(t)$ denoting the j-th row. These are parameterized by $\alpha(t)$ and $\beta(t)$:

$$Z_i(t) \sim \mathcal{M}(1, \alpha(t)), \quad W_j(t) \sim \mathcal{M}(1, \beta(t)),$$

where $\mathcal{M}(\cdot, \cdot)$ is the multinomial probability mass function. $\alpha_q(t) = \mathbb{P}\{z_{iq}(t) = 1\}$ and $\beta_\ell(t) = \mathbb{P}\{w_{j\ell}(t) = 1\}$, with $\sum_{q=1}^{Q} \alpha_q(t) = 1$ and $\sum_{\ell=1}^{L} \beta_\ell(t) = 1$. Z and W are assumed to be independent.

Sparsity modeling

In order to model a potentially extreme sparsity, the observed data are assumed to follow a mixture of block-conditional zero-inflated Poisson (ZIP) distributions, where

the entries $X_{ij}(t)$ are conditionally independent in (i, j, t):

$$X_{ij}(t)|Z_i(t), W_j(t) \sim ZIP(\Lambda_{Z_i(t),W_j(t)}, \pi(t)), \tag{1}$$

where Λ is the $Q \times L$ block-dependent intensity function of the Poisson component, and $\pi(t)$ is a vector of length T that indicates the level of sparsity at any given time period. In order to ease the inference, we finally provide an equivalent formulation of the above equations in terms of a hidden random matrix, $A \in \{0, 1\}^{N \times M}$, where, independently for all i and j:

$$A_{ij}(t) \sim \mathcal{B}(\pi(t)),$$

with $\mathcal{B}(\cdot)$ denoting the Bernoulli distribution of parameter $\pi(t)$ and such that

$$\begin{aligned} &A_{ij}(t) = 1 \Rightarrow X_{ij}(t)|Z_i(t),\ W_j(t) = 0, \\ &A_{ij}(t) = 0 \Rightarrow X_{ij}(t)|Z_i(t),\ W_j(t) \sim \mathcal{P}(X_{ij}(t), \Lambda_{Z_i(t),W_j(t)}), \end{aligned} \tag{2}$$

where $\mathcal{P}(\cdot, \Lambda)$ denotes the Poisson distribution with intensity parameter Λ.

Modeling the temporal evolution of the parameters

We assume that the evolution of the mixing proportions α, β, and the sparsity parameter π are governed by a system of ordinary differential equations (ODEs). By using ODEs, we can model the temporal evolution of both the composition of clusters and sparsity. Since we work in discrete time we discretize the dynamic systems by making use of their Euler schemes:

$$\begin{cases} a(t+1) = a(t) + f_Z(a(t)), \\ b(t+1) = b(t) + f_W(b(t)), \\ c(t+1) = c(t) + f_A(c(t)), \end{cases} \qquad \begin{cases} \alpha_q(t) = \text{softmax}(a_q(t)) = e^{a_q(t)} / \sum_{q=1}^{Q} e^{a_q(t)}, \\ \beta_\ell(t) = \text{softmax}(b_\ell(t)) = e^{b_\ell(t)} / \sum_{\ell=1}^{L} e^{b_\ell(t)}, \\ \pi(t) = e^{c(t)}/1 + e^{c(t)}. \end{cases}$$

where f_Z, f_W and f_A are assumed to be three continuously differentiable functions.

Remark that, if we want to relax the condition of equally spaced points, we can introduce a parameter Δ_t such that

$$a(t + \Delta_t) \cong a(t) + f_Z(a(t)) \cdot \Delta_t,$$

where Δ_t indicates the length of the considered time interval. Henceforth, in order to simplify the exposition, we assume that Δ_t is constant, denoted as Δ. Furthermore, for convenience, we set $\Delta = 1$ without loss of generality.

2.2. The joint distribution

The set of the model parameters is denoted by $\theta = (\Lambda, \alpha, \beta, \pi)$ and the latent variables used so far are Z, W, and A, where we denote $\alpha \triangleq \{\alpha(t)\}_t$, $\beta \triangleq \{\beta(t)\}_t$, $\pi \triangleq \{\pi(t)\}_t$. Thus, the likelihood of the complete data reads

$$p(X, Z, W, A|\theta) = p(X|Z, W, A, \Lambda, \pi)p(A \mid \pi)p(Z|\alpha)p(W|\beta). \tag{3}$$

The terms on the right hand side of the above equation can be further developed, as in Marchello et al. (2024).

3. Online Inference for Stream Data

In this section, we present the online extension of the Zip-dLBM method, called Stream Zip-dLBM. The objective is to perform co-clustering of rows and columns in real-time as new data become available. To prevent memory overload, we have revisited the original inference algorithm of Zip-dLBM, enabling the data to be processed without the need to store it in memory. To allow the algorithm to update the parameter estimates continuously as a new data is incorporated, we use a moving window, $G_d(t)$, of size d. In more detail, at time t, we keep in memory only the data in the interval $[t-d, t]$, namely $X(t-d), X(t-d+1), \ldots, X(t)$, that will be used for the estimation of the model parameters. The data outside the interval can be discarded to prevent memory overloads and maintain the algorithm's functionality. Once a time point t quit the time window (after passing through it) the parameter estimates at that point become fixed, and the "past" data can be discarded by the inference procedure.

3.1. Variational assumption

To estimate model parameters, traditional methods maximize the log-likelihood $p(X|\theta)$. However, direct maximization and the EM-algorithm are infeasible due to the complex dependencies and continuous evolution of latent variables (Z, W). Additionally, α, β, and π cannot be updated using closed formulas due to their link with systems of ordinary differential equations.

Instead, we use a combination of Variational-EM inference and Stochastic Gradient Descent (SGD). We introduce a variational distribution $q(.)$ over (A, Z, W) to decompose the observed data log-likelihood as

$$\log p(X|\theta) = \mathcal{L}(q, \theta) + KL(q(.)||p(.|X, \theta)), \tag{4}$$

where $\mathcal{L}$, the lower bound of $p(X|\theta)$, is

$$\mathcal{L}(q, \theta) = E_{q(A,Z,W)}[\log(p(X, A, Z, W|\theta)] - E_{q(A,Z,W)}[\log(q(A, Z, W))]. \tag{5}$$

The Kullback-Liebler divergence KL between the true and approximate posterior $q(\cdot)$ is

$$KL(q(.)||p(.|X, \theta)) = -\sum_A \sum_Z \sum_W q(A, Z, W) \log \frac{p(A, Z, W|X, \theta)}{q(A, Z, W)}. \tag{6}$$

Our goal is to find a distribution $q(.)$ that maximizes the lower bound $\mathcal{L}(q, \theta)$. To optimize $\mathcal{L}(q, \theta)$, we use the mean field approximation, assuming $q(A, Z, W)$ factorizes as follows for all t:

$$q(A(t), Z(t), W(t)) = q(A(t))q(Z(t))q(W(t)) = \prod_{i=1}^{N} \prod_{j=1}^{M} q(A_{ij}(t)) \prod_{i=1}^{N} q(Z_i(t)) \prod_{j=1}^{M} q(W_j(t)). \tag{7}$$

3.2. Variational E-step

The optimal variational updates of $q(\cdot)$, under the assumption in Eq. (7), can be obtained as in Bishop (2008, Ch.10). Denoting by $\delta_{ij}(t) := q(A_{ij}(t) = 1)$ the variational

probability of success for $A_{ij}(t)$, the optimal update is

$$\delta_{ij}(t) = \frac{\exp(R_{ij}(t))}{(1+\exp(R_{ij}(t)))}, \tag{8}$$

with

$$R_{ij}(t) = \log(\pi(t)\mathbf{1}_{\{X_{ij}(t)=0\}}) + \sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big[- E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]X_{ij}(t)\log\Lambda_{q\ell}$$
$$+ E_{q(W,Z)}[Z_{iq}(t)]E_{q(W,Z)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big] + \log X_{ij}(t)! - \log(1-\pi(t)).$$

Note that, formally, when $X_{ij}(t) \neq 0$, $R_{ij}(t) = -\infty$ and $\delta_{ij}(t) = 0$, which makes sense: non-null observations in X come from a Poisson distribution with probability one (see Eq. (2)).

Regarding $q(Z)$, let us denote by $\tau_{iq}(t) := q(Z_{iq}(t) = 1)$ the variational probability of success of $Z_{iq}(t)$. The optimal update can be written as

$$\tau_{iq}(t) = \frac{r_{iq}(t)}{\sum_{q_0=1}^{Q} r_{iq_0}(t)}, \tag{9}$$

with $r_{iq}(t)$ is denoted by

$$r_{iq}(t) \propto \exp\left(\sum_{j=1}^{M}\sum_{\ell=1}^{L}\Big\{(1-E_{q(A,W)}[A_{ij}(t)])\Big[E_{q(A,W)}[W_{j\ell}(t)]X_{ij}(t)\log(\Lambda_{q\ell})\right.$$
$$\left. - E_{q(A,W)}[W_{j\ell}(t)]\Lambda_{q\ell}\Big]\Big\} + \log(\alpha_q(t))\right).$$

Similarly for the latent variable W, denoting by $\eta_{j\ell}(t) := q(W_{j\ell}(t) = 1)$ the variational probability for $W_{j\ell}(t)$, the optimal update of $q(W)$ is

$$\eta_{j\ell}(t) = \frac{s_{j\ell}(t)}{\sum_{\ell_o=1}^{L} s_{j\ell_o}(t)}, \tag{10}$$

where

$$s_{j\ell}(t) \propto \exp\left(\sum_{i=1}^{N}\sum_{q=1}^{Q}\Big\{(1-E_{q(A,Z)}[A_{ij}(t)])\Big[E_{q(A,Z)}[Z_{iq}(t)]X_{ij}(t)\log(\Lambda_{q\ell})\right.$$
$$\left. - E_{q(A,Z)}[Z_{iq}(t)]\Lambda_{q\ell}\Big]\Big\} + \log(\beta_\ell(t))\right).$$

The proofs of Equations (8), (9) and (10) are provided in Marchello et al. (2024). It is worth noting that these update equations can be executed step by step, allowing for incremental updates of the variational parameters. Also, note that the update in these equations can be computed independently for any pair (i, j), at any time point t.

3.3. Online variational M-step

While the updates in the E-step for $\tau(t)$, $\eta(t)$, and $\delta(t)$ depend solely on the current time instant t, the same cannot be said for the updates in the M-step. The M-step involves updating the model parameters, such as $\theta = (\Lambda, \alpha, \beta, \pi)$, based on the current estimates obtained in the E-step. In order to obtain the updates of the parameter set θ, the objective of the M-Step is the maximization of the lower bound $\mathcal{L}(q, \theta)$ with respect to $\theta = (\Lambda, \alpha, \beta, \pi)$, while holding the variational distribution $q(\cdot)$ fixed. Denoting t as the current time instant, we develop Eq. (5) such that the variational lower bound $\mathcal{L}(q, \theta)$ can be written as

$$
\begin{aligned}
\mathcal{L}(q,\theta) = & \sum_{u=1}^{t}\sum_{i=1}^{N}\sum_{j=1}^{M}\Bigg\{\delta_{ij}(u)\log(\pi(u)\mathbf{1}_{\{X_{ij}(u)=0\}}) + (1-\delta_{ij}(u))\Bigg[\log(1-\pi(u)) \\
& + \sum_{q=1}^{Q}\sum_{\ell=1}^{L}\Big\{\tau_{iq}(u)\eta_{j\ell}(u)X_{ij}(u)\log\Lambda_{q\ell} - \tau_{iq}(u)\eta_{j\ell}(u)\Lambda_{q\ell}\Big\}\Bigg] \\
& - (1-\delta_{ij}(u))\log(X_{ij}(u)!)\Bigg\} \\
& + \sum_{u=1}^{t}\sum_{i=1}^{N}\sum_{q=1}^{Q}\tau_{iq}(u)\log(\alpha_q(u)) + \sum_{u=1}^{t}\sum_{j=1}^{M}\sum_{\ell=1}^{L}\eta_{j\ell}(u)\log(\beta_\ell(u)) \\
& - \sum_{u=1}^{t}\sum_{i=1}^{N}\sum_{q=1}^{Q}\tau_{iq}(u)\log(\tau_{iq}(u)) - \sum_{u=1}^{t}\sum_{j=1}^{M}\sum_{\ell=1}^{L}\eta_{j\ell}(u)\log(\eta_{j\ell}(u)) \\
& - \sum_{u=1}^{t}\sum_{i=1}^{N}\sum_{j=1}^{M}\Big(\delta_{ij}(u)\log(\delta_{ij}(u)) + (1-\delta_{ij}(u))\log(1-\delta_{ij}(u))\Big).
\end{aligned}
\tag{11}
$$

Update of Λ

Here our goal is to derive the online update of the zero-inflated Poisson intensity parameter, Λ. The variational distribution $q(A, Z, W)$ is kept fixed, while the lower bound is maximized with respect to Λ at every time instant t, to obtain its update, $\hat{\Lambda}$.

The updating formula of Λ is obtained by maximizing $\mathcal{L}(q, \theta)$ with respect to the parameter. Denoting $\hat{\Lambda}_{q\ell}^{old} = N_{q\ell}^{old}/D_{q\ell}^{old}$, we obtain the final online update

$$
\hat{\Lambda}_{q\ell} = \hat{\Lambda}_{q\ell}^{old}\cdot\frac{D_{q\ell}^{old}}{D_{q\ell}^{old}+D_{q\ell}^{(t)}} + \frac{N_{q\ell}^{(t)}}{D_{q\ell}^{old}+D_{q\ell}^{(t)}},
\tag{12}
$$

with

- $N_{q\ell}^{(t)} = \sum_{i=1}^{N}\sum_{j=1}^{M}\tau_{iq}(t)\eta_{j\ell}(t)(X_{ij}(t) - \delta_{ij}(t)X_{ij}(t))$,
- $D_{q\ell}^{old} = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{u=1}^{(t-1)}\tau_{iq}(u)\eta_{j\ell}(u)(1-\delta_{ij}(u))$,
- $D_{q\ell}^{(t)} = \sum_{i=1}^{N}\sum_{j=1}^{M}\tau_{iq}(t)\eta_{j\ell}(t)(1-\delta_{ij}(t))$.

Hence, when a new observation comes, Λ can be updated thanks to Eq.(17), along with the update of $D_{q\ell}^{old}$. The proof is provided in Appendix A.1.

Update of α, β and π through deep neural networks

As mentioned in Section 2.1.3, the mixture proportions α and β, as well as the sparsity parameter π are driven by three systems of differential equations, respectively. Hence, the update process for these parameters in the online inference algorithm poses a challenge because they lack closed-form updating formulas. As a result, the decomposition strategy used for updating Λ cannot be applied. To address this issue, we introduce an approximation technique that leverages a moving window $G_d(t)$ of size d, allowing us to update the parameters based on the most recent d observations. In addition to its role in parameter updates, the moving window $G_d(t)$ serves another purpose as the input for a deep neural network. As we assumed that the functions f_A, f_W and f_Z are continuous, we propose to parametrize them with three LSTM networks (Hochreiter and Schmidhuber 1997). LSTM is a type of recurrent neural network that operates on sequences of a specific length and produces a sequence of the same length, shifted one time step ahead. For instance, let's consider the current time t and the time window $G_d(t)$ with a length of d. The input for the LSTM networks consists of a series of values ranging from $t-1-d$ to $t-1$, representing the historical observations within the window. The LSTM networks then predict a sequence of values from $t-d$ to t, which correspond to the updated parameter values for α, β, and π. Optimizing the lower bound in Eq.(11) with respect to α, β, and π reduces to maximize it with respect to the parameters of the neural networks. For instance, the loss function for α can be expressed as follows

$$\mathcal{L} = \sum_{u \in G_d(t)} \sum_{i=1}^{N} \sum_{q=1}^{Q} \tau_{iq}(u) \log \alpha_q(u). \tag{13}$$

The loss functions of β and π can be similarly derived using their respective distribution-specific equations. The whole inference procedure is summarized in Algorithm 1 in the Appendix.

3.4. Initialization and model selection

In clustering methods based on the EM algorithm, initializing and selecting the appropriate number of clusters for rows and columns are crucial. These tasks become more complex when using deep neural networks to model cluster dynamics and sparsity, but these networks also provide flexibility that reduces computational burdens, as shown in Section 4.2. Here, we modify the initialization method from Marchello et al. (2024) for online adaptation. First, we apply a static LBM algorithm to the initial data slice X_{t_0} for various cluster pairs (q, ℓ), where $q = 2, \ldots, Q_{max}$ and $\ell = 2, \ldots, L_{max}$. The ICL criterion (Integrated Completed Likelihood, Biernacki et al. (2000)) determines the optimal cluster numbers

$$\begin{aligned} ICL(Q, L) = &\log p(X, \hat{Z}, \hat{W}; \hat{\theta}) - \frac{Q-1}{2} \log N - \frac{L-1}{2} \log M \\ &- \frac{QL}{2} \log(NM) - \frac{1}{2} \log(NM). \end{aligned} \tag{14}$$

The pair $(\hat{Q}, \hat{L})$ that leads to the highest value for the ICL is considered as the most meaningful cluster numbers for the considered slice of data X_{t_0}. The pair $(\hat{Q}, \hat{L})$ with the highest ICL is considered optimal for X_{t_0}. However, since $\hat{Q}$ and $\hat{L}$ may not be optimal

for all future times, the VEM-SGD algorithm runs with more components ($Q_{max} \geq \hat{Q}$ and $L_{max} \geq \hat{L}$). New data entries initialize part of the model parameters with $\hat{\theta}(t)$ from a static LBM run, while additional clusters start at zero. Deep neural networks allow VEM-SGD to initialize with empty clusters, placeholders for new patterns. LSTMs capture complex data relationships even with sparse data, focusing computation on relevant data parts and delaying empty cluster processing. This avoids the need to run the algorithm with all cluster combinations, handling large datasets efficiently as demonstrated in the next section. Initially, for the first d time points, $\alpha(t)$, $\beta(t)$, and $\pi(t)$ are modeled via two-layer fully connected neural networks following Marchello et al. (2024). After $t = d$, estimates from the previous step serve as LSTM inputs for online parameter estimation. More details are in Section 3.3 and Appendix A.4.

3.5. Bayesian online change point detection

As previously stated, one of the aims of Stream Zip-dLBM is to perform multivariate online change point detection. To accomplish this task, we combine the Bayesian Online Change Point Detection (BOCD) method, proposed in a seminal paper by Adams and MacKay (2007), with our strategy. BOCD detects change points based on the estimation of the posterior distribution over the current "run length", or time segment since the last change point, given the data observed so far, using a simple message-passing algorithm. Essentially, the run length is used to determine if a new data point belongs to the current partition based on previous observations. If the new data point belongs to the current partition, the run length will increase by 1 at the next time step, otherwise it will reset to 0. This process is continuously repeated at each time step. It is worth noticing that the BOCD algorithm is typically implemented in an online fashion, analyzing the data as it streams in. However, in our case, we directly apply the algorithm to detect change points on the estimates of $\alpha(t)$, $\beta(t)$, and $\pi(t)$ that are generated by the LSTM. To prevent detecting change points on parameters that will be recalculated in future time steps, we run the BOCD algorithm only on time points "behind" $G_d(t)$. Stated differently, at time t, BOCD operates on parameter values at time instances $t - d$.

4. Numerical Experiments

This section highlights the key features of Stream Zip-dLBM using simulated datasets and validates the inference algorithm described earlier. The first experiment applies Stream Zip-dLBM to a dataset with evolving block patterns and sparsity, showing its capability to recover data structure in real-time. The second experiment demonstrates the model selection procedure on a simulated dataset.

4.1. Introductory example

A simulated dataset with dimension $350 \times 300 \times 150$ has been generated according to our model to perform this experiment. The simulated dynamics of α, β and π can be seen on the left-hand side of Figure 1. Concretely, α, β and π are three time series independently fluctuating around constant trends. Fluctuations are obtained at each time by means of independent Gaussian distributions with constant variance. The mean levels change

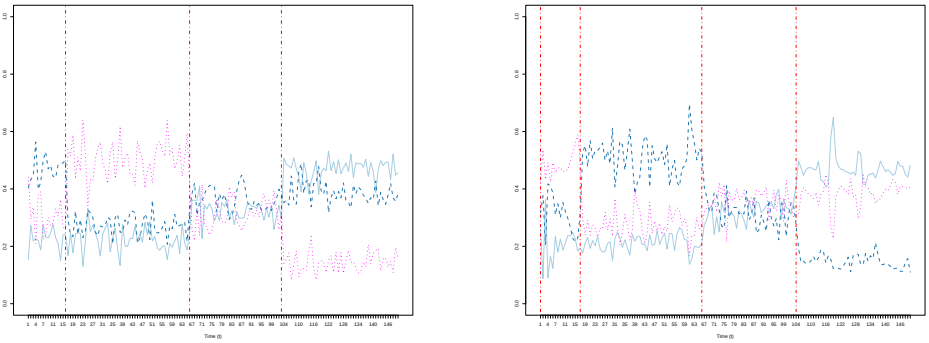
when a change point occurs. The levels of the simulated change points and the values of the simulated parameter Λ are indicated in the Appendix 5. Based on the mixture proportions α, β, the values of the latent variables were then simulated through their distributions. Next, we used the sparsity proportions, π, and the intensity parameter, Λ, to simulate the three-dimensional tensor X as zero-inflated Poisson variables. We then applied the Stream Zip-dLBM inference algorithm to the simulated dataset, using the actual values of $Q = 3$ and $L = 2$ to demonstrate the model's ability to recover the parameters. Figure 1 displays the true mixture proportions on the left side and the online estimates on the right side. The red dashed lines depict the simulated and estimated change points, respectively. From these results we see that Stream Zip-dLBM perfectly recovers the evolution of the mixing proportion and the sparsity parameter over time, including the change points.

4.2. Model selection experiment

In this experiment, we assess the ability of Stream Zip-dLBM to determine the optimal number of clusters for rows and columns. Initially, we utilize the Integrated Completed Likelihood (ICL) criterion to compute the optimal number of clusters on the first slice. This ensures that the algorithm is initialized with the best possible parameters. Once initialized, the algorithm maintains consistent results without making any alterations, thus retaining the optimal number of clusters. To evaluate the effectiveness of the algorithm and verify if it activates new clusters, we generate a dataset based on the configuration described in Section 4.1. The dataset consists of 3 row clusters (Q=3) and 2 column clusters (L=2). We then apply Stream Zip-dLBM to this dataset, with maximum values of $Q_{max} = 7$ and $L_{max} = 7$. Figure 2 provides an illustrative demonstration of the algorithm's behavior, specifically regarding the activation of clusters. It is clear from the figure that the unnecessary clusters remain empty and that the estimates of the α, β, and π parameters are also accurate. Finally, it is worth noticing that Stream Zip-dLBM successfully identifies the changing points in α, β and π over time, despite not using the optimal number of input clusters. Finally, to evaluate the performance of the model in identifying the correct rows and columns partitions, we use the adjusted Rand index (ARI) (Rand 1971). The adjusted Rand index, from a mathematical point of view, is closely related to the accuracy measure, however it is a commonly used method for evaluating clustering problems since it can be applied for measuring the similarity between two partitions even with different number of clusters and it is invariant to label switching. We also use a measure called CARI, recently introduced by Robert et al. (2021). This new criterion is based on the Adjusted Rand Index (Rand 1971) and it was developed especially for being applied to co-clustering methods. The closer these indexes are to 1, the more two label vectors are similar to each other. We compared the original matrices Z and W, with the estimates τ and η given by the output of Stream Zip-dLBM. We evaluate the performance indexes at each time step, obtaining the following results:

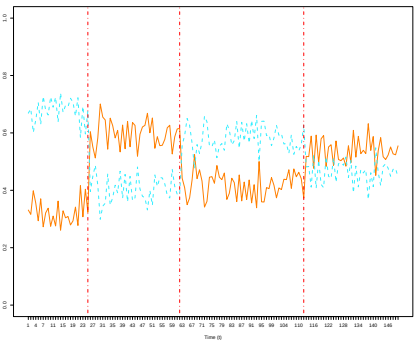
Table 1:

ARI rows	ARI columns	CARI
0.99 ± 0.03	1 ± 0	0.99 ± 0.02

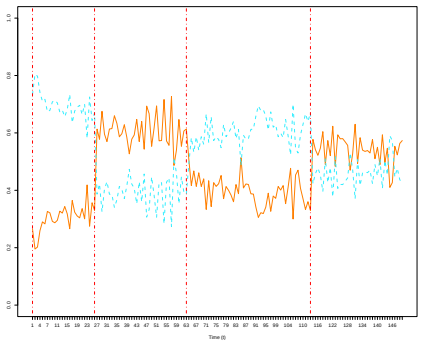


(a) True α.

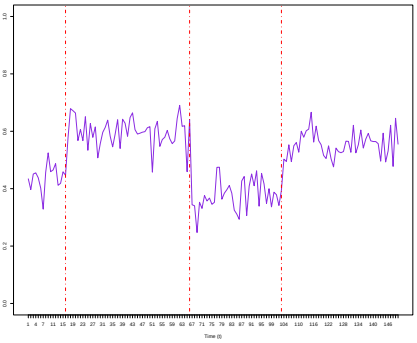
(b) Estimated α.



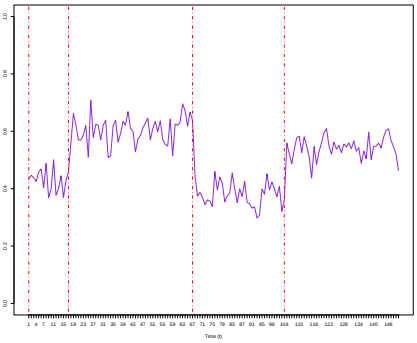
(c) True β.



(d) Estimated β.



(e) True π.



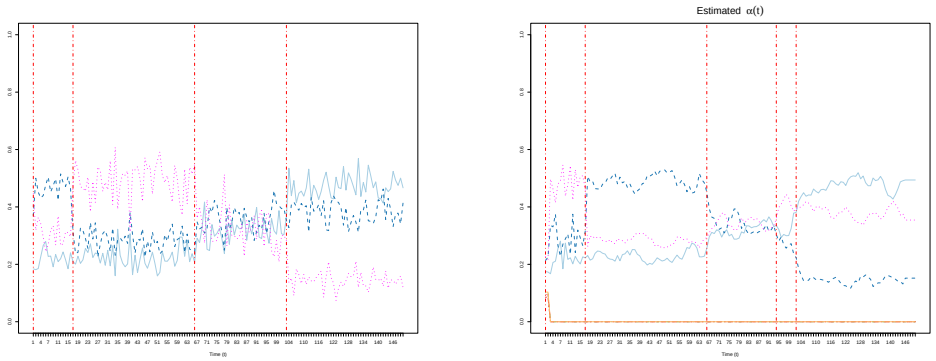
(f) Estimated π.

Figure 1: Evolution of the true (left) and estimated (right) proportions of the parameters α, β and π, respectively.

Thus, we can conclude that our algorithm satisfyingly identifies the composition of the original clusters in time.

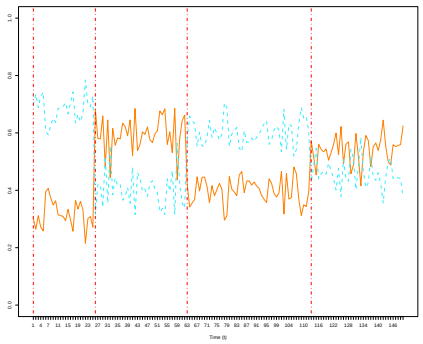
5. Analysis of the Adverse Drug Reaction Dataset

This section focuses on the online application of Stream Zip-dLBM to a large-scale pharmacovigilance dataset, with the aim of illustrating the potential of the tool for such studies.

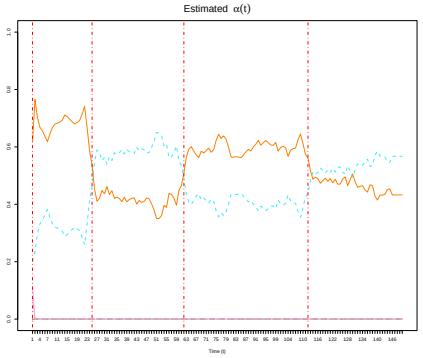


(a) True α.

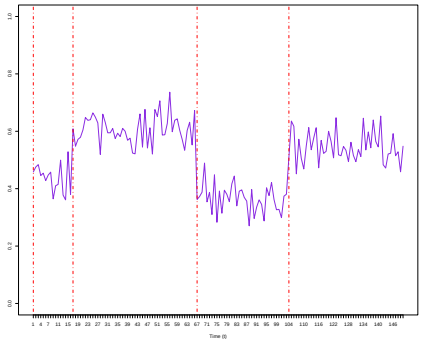
(b) Estimated α.



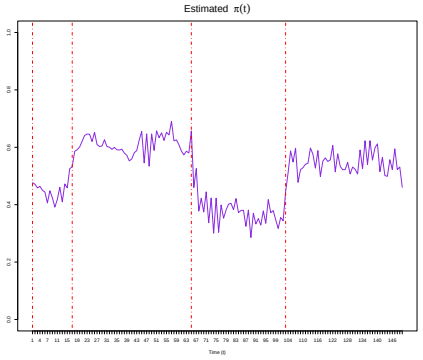
(c) True β.



(d) Estimated β.



(e) True π.

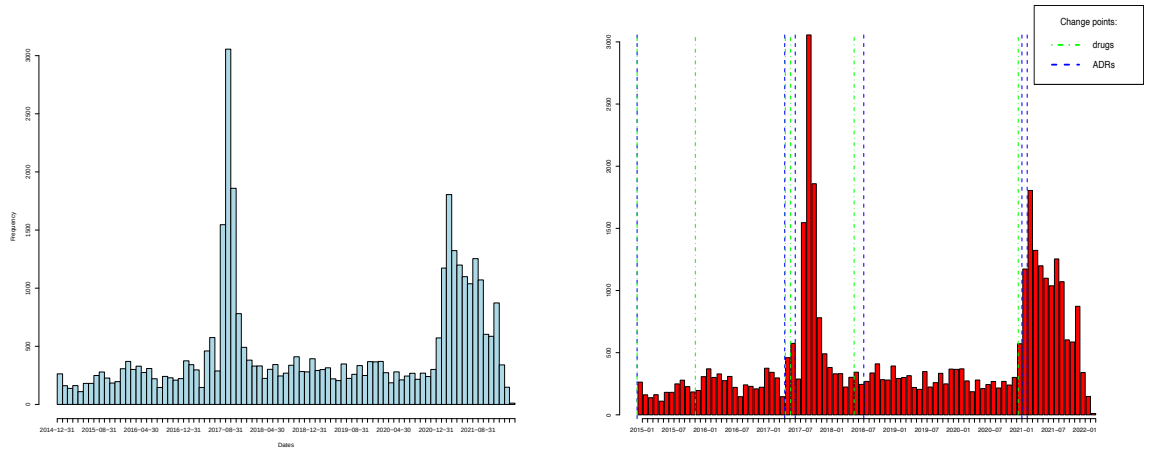


(f) Estimated π.

Figure 2: Evolution of the true (left) and estimated (right) proportions of the parameters α, β and π, respectively.

5.1. Protocol and data

This section examines a large adverse drug reaction (ADR) dataset collected by the Regional Center of Pharmacovigilance (RCPV) at the University Hospital of Nice, covering an area with 2.3 million inhabitants over 7 years, from January 1^{st}, 2015, to March 3^{rd}, 2022. Due to data sparsity, monthly aggregations were made, resulting in a dataset with 39,267 declarations, including drug names, ADRs, and reception dates. We focused on drugs and ADRs reported more than 10 times, yielding 419 drugs, 614 ADRs, and 87 time intervals with 23,264 non-zero entries. To avoid duplicate entries for the same molecule under different brand names, we used international nonproprietary names (INNs). Looking at Figure 3a, in 2017 we notice a significant increase in ADR



(a) Number of reports by month.

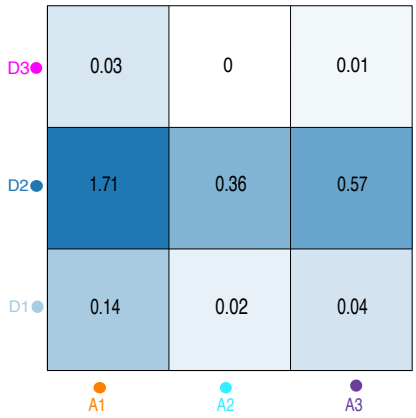
(b) Number of reports by month with change points.

Figure 3: On the left, number of declarations received by the pharmacovigilance center from 2015 to 2022, sorted by month. On the right, histogram of declarations over time, with change points.

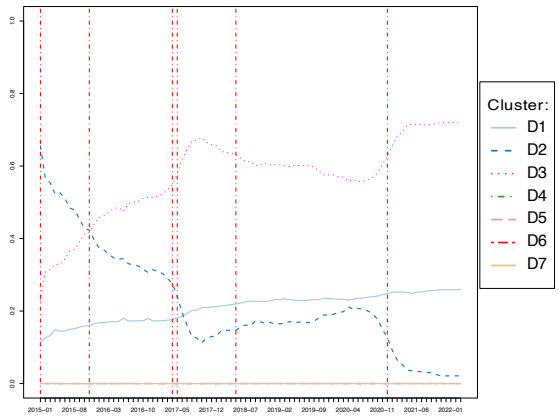
reports for Lévothyrox®, used to treat hypothyroidism. This peak was due to a new formula introduced to correct drug stability issues, which received extensive media coverage. Additionally, since late 2020, COVID-19 vaccinations began, with Comirnaty®, Moderna®, and Vaxzevria® being introduced sequentially. The dataset reflects varying signal amplitudes, highlighting both prominent and less visible ADRs, such as those involving Mirena® in 2017, which also caused health policy concerns. Stream Zip-dLBM is expected to help uncover hidden signals within this sparse dataset, which has a sparsity range of 99.25% to 99.98% per month. To prevent numerical issues, an LSTM network inferred the parameters α and β, with point estimates of $\hat{\pi}$ used in the inference process.

5.2. Summary of the results

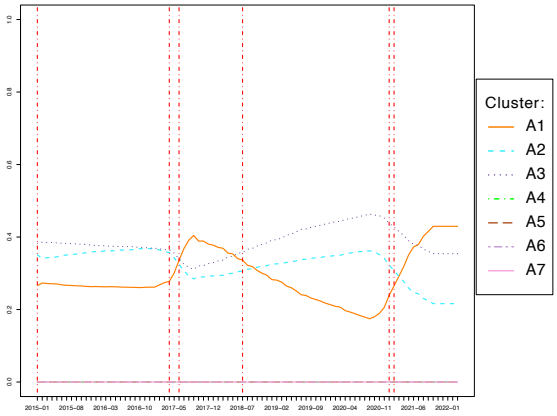
To initialize the algorithm, we computed the ICL criterion on the first month's data, identifying the optimal clusters as $\hat{Q} = 3$ and $\hat{L} = 3$ (Section 3.4). We then ran Stream Zip-dLBM, updating parameters with each new entry in tensor X, with $Q_{max} = 7$ and $L_{max} = 7$ to allow dynamic clustering. The process took about an hour on a 2020 MacBook Pro, with a 2.3 GHz Quad-Core Intel Core i7 and 16 GB RAM, using a 5-month moving window $G_d(t)$. In Figure 3b, shows monthly declaration frequencies from 2015 to 2022, with detected change points marked by dashed lines: green for drug clusters and blue for ADR clusters.Figure 4a presents estimated Poisson intensities (Λ) for 3 drug clusters (D) and 3 ADR clusters (A). This figure only focuses on the 3 groups of drug clusters and the 3 groups of ADR clusters that have been activated in the inference. This representation provides valuable insights for model interpretation as it gives an overview of the relationships between drug clusters and ADR clusters and how they evolve over time. Each color refers to a drug (rows) or ADR (columns) cluster and the higher is the value in each block, the strongest is the relationship (i.e the expected number of declarations received in the time unit) between the related



(a) Estimated Poisson intensity parameter.



(b) Evolution of drug clusters proportions.



(c) Evolution of ADR clusters proportions.

(d) Evolution of the estimates $\hat{\pi}$.

Figure 4: Estimated Poisson intensities, each color represents a different drug (ADR) cluster (top left); evolution of the estimates of $\hat{\alpha}$ (top right); evolution of the estimates of $\hat{\beta}$ (bottom left), evolution of the estimates $\hat{\pi}$ (bottom right).

pair of clusters. Figures 4b and 4c display the estimated mixture proportions of drug clusters ($\hat{\alpha}$) and ADR clusters ($\hat{\beta}$) respectively. The dashed lines in the figures represent the change points identified by the BOCD algorithm. Figure 4d illustrates the sparsity parameter $\hat{\pi}$, showing no change points due to insufficient variability, as the values range goes from 0.99 to 1. Figures 4a, 4b, and 4c reveal that the clusters with the highest intensity are the least populated. For instance, cluster drug cluster D2 exhibits a significant level of interaction with ADR clusters A1 and A3. However, from May 2017. D2's population declines significantly. This phenomenon is attributed to the presence of drugs associated with significant health crises that happened during the reporting period. Notably, Mirena® in early 2017, Lévothyrox® in late 2017, and Covid-19 vaccines in 2021 were the primary drivers of these crises. Each crisis period corresponds to a detected change point in both drug and ADR cluster proportions. Similarly, analyzing clusters A1 and A3 reveals the most reported ADRs during each crisis. Cluster A1 includes mainly hormonal side effects (e.g., anxiety, heat shock, aggressive behavior) during the Mirena® crisis. During the Lévothyrox® crisis, A1 has

a peak, likely due to extensive media coverage prompting diverse side effect reports. In 2021, A1 peaks with Covid-19 vaccine-related ADRs, such as arm pain, inflammation, and flu-like symptoms. Cluster D1 remains empty until August 2017, then includes widely used drugs like paracetamol and amoxicillin, frequently reported after the Lévothyrox® crisis. The evolution of cluster D1's proportion, shown in Figure 4b, aligns with the change points identified by the algorithm. Cluster D1's evolution, depicted in Figure 4b, shows significant changes corresponding to detected change points. Despite initial emptiness, D1 begins to include widely used drugs such as paracetamol and amoxicillin after the Lévothyrox® crisis, indicating a shift in reporting patterns. In contrast, Cluster D3 maintains consistently low interaction intensities with ADR clusters, as illustrated in Figure 4b. Post-2017, D3 comprises drugs with low reporting frequencies, reflecting its distinct role in the pharmacovigilance landscape. Examining Figures 4a and 4c, clusters A2 and A3 initially house the majority of adverse effects. However, during the Mirena® crisis, A2 and A3 retain only residual effects unrelated to the specific health issue, while cluster A1 becomes predominant. This shift intensifies after October 2017, with A1 focusing specifically on Lévothyrox®-related ADRs such as hair loss and insomnia. After the Lévothyrox® crisis, cluster A1 becomes empty until the subsequent change point detected in January 2021. From this moment until the peak of Covid-19 vaccine reports in February 2022, cluster A1 includes the main adverse effects reported for Covid-19 vaccines (e.g. pain at the vaccination site, skin rash, pericarditis, etc.). Also, Figure 4c clearly highlights the Lévothyrox® crisis as a pivotal moment in pharmacovigilance history, likely prompting increased reporting of drug and vaccine side effects. Lastly, Figure 4d shows the estimated evolution of the sparsity parameter, indicating minimal variability and no detected change points in $\hat{\pi}$. Initially high at 99.7% in 2015, sparsity decreases to 98.75% by the peak in 2017 and reaches a global minimum of 98.65% in March 2021, coinciding with the Covid-19 vaccine reporting peak. The Stream Zip-dLBM algorithm performed very well on the pharmacovigilance dataset, providing meaningful a segmentation on drugs and adverse effect clusters over time. Its ability to adapt and process data in real-time ensured accurate and insightful analysis, proving highly effective for monitoring and identifying safety signals.

6. Conclusions

This paper addresses the need to analyze and summarize observations and features of a dynamic matrix in an online setting for pharmacovigilance. We propose an online dynamic co-clustering technique that clusters rows and columns over time, allowing for changes in cluster memberships and detecting structural changes in interactions. Our method introduces a generative zero-inflated dynamic latent block model as an online extension of Zip-dLBM, using three systems of ordinary differential equations for time modeling. Inference is done via a Variational EM algorithm and stochastic optimization of LSTM network parameters. We also incorporate an online change point detection method to create real-time alerts. The approach is evaluated with simulated data scenarios and applied to a large dataset from the Regional Center of Pharmacovigilance of Nice (France), providing meaningful online segmentation of drugs and adverse drug reactions. This model shows promise for identifying significant pharmacovigilance pat-

terns or emerging public health concerns. We are currently developing a web platform for the Regional Center of Pharmacovigilance in Nice to run the model on incoming data, identify structural changes, and send email notifications with reports. The software's effectiveness will be assessed over six months, with plans for national expansion. The online inference algorithm and change point detection enable Stream Zip-dLBM to continuously analyze ADR data and trigger alerts, facilitating further investigation and intervention by medical authorities.

Acknowledgement

This work has been supported by the French government, through the 3IA Côte d'Azur, Investment in the Future, project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002. We would like to extend our gratitude to the Regional Center of Pharmacovigilance (RCPV) of Nice for providing the data and the invaluable support throughout this research.

References

Adams, R. P. and MacKay, D. J. (2007). Bayesian online changepoint detection. https://arxiv.org/abs/0710.3742.

Bergé, L. R., Bouveyron, C., Corneli, M., and Latouche, P. (2019). The latent topic block model for the co-clustering of textual interaction data. *Computational Statistics & Data Analysis*, 137:247–270. DOI: 10.1016/j.csda.2019.03.005.

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. DOI: 10.1109/34.865189.

Biernacki, C., Jacques, J., and Keribin, C. (2023). A survey on model-based co-clustering: High dimension and estimation challenges. *Journal of Classification*, 40(2):332–381. 10.1007/s00357-023-09441-3.

Bishop, C. M. (2008). *Pattern Recognition and Machine Learning*. Springer. ISBN: 0387310738.

Boutalbi, R., Labiod, L., and Nadif, M. (2020). Tensor latent block model for co-clustering. *International Journal of Data Science and Analytics*, 10, 161–175. DOI: 10.1007/s41060-020-00205-5.

Boutalbi, R., Labiod, L., and Nadif, M. (2021). Implicit consensus clustering from multiple graphs. *Data Mining and Knowledge Discovery*, 35(6):2313–2340. DOI: 10.1007/s10618-021-00788-y.

Bouveyron, C., Bozzi, L., Jacques, J., and Jollois, F.-X. (2018). The functional latent block model for the co-clustering of electricity consumption curves. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 67(4):897–915. DOI: 10.1111/rssc.12260.

Bouveyron, C., Celeux, G., Murphy, T. B., and Raftery, A. E. (2019). *Model-Based Clustering and Classification for Data Science: With Applications in R*. Cambridge University Press. ISBN: 978-1-108-49420-5.

Casa, A., Bouveyron, C., Erosheva, E., and Menardi, G. (2021). Co-clustering of time-dependent data via the shape invariant model. *Journal of Classification*, 38(3):626–649. DOI: 10.1007/s00357-021-09402-8.

Corneli, M., Bouveyron, C., and Latouche, P. (2020). Co-clustering of ordinal data via latent continuous random variables and not missing at random entries. *Journal of Computational and Graphical Statistics*, 29(4):771–785. DOI: 10.1080/10618600.2020.1739533.

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274. DOI: 10.1145/502512.502550.

Dhillon, I. S., Mallela, S., and Modha, D. S. (2003). Information-theoretic co-clustering. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 89–98. DOI: 10.1145/956750.956764.

Ding, C., Li, T., Peng, W., and Park, H. (2006). Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 126–135. DOI: 10.1145/1150402.1150420.

Govaert, G. and Nadif, M. (2003). Clustering with block mixture models. *Pattern Recognition*, 36(2):463–473. DOI: 10.1016/S0031-3203(02)00074-2.

Govaert, G. and Nadif, M. (2010). Latent block model for contingency table. *Communications in Statistics - Theory and Methods*, 39(3):416–425. DOI: 10.1080/03610920903140197.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780. DOI: 10.1007/978-3-642-24797-2_4.

Jacques, J. and Biernacki, C. (2018). Model-based co-clustering for ordinal data. *Computational Statistics & Data Analysis*, 123:101–115. DOI: 10.1016/j.csda.2018.01.014.

Kavitha, V. and Punithavalli, M. (2010). Clustering time series data stream-a literature survey. DOI: 10.48550/arXiv.1005.4270.

Kawahara, Y. and Sugiyama, M. (2009). Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 389–400. SIAM. DOI: 10.1137/1.9781611972795.34.

Keribin, C., Brault, V., Celeux, G., and Govaert, G. (2015). Estimation and selection for the latent block model on categorical data. *Statistics and Computing*, 25(6):1201–1216. DOI: 10.1007/s11222-014-9472-2.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. DOI: 10.48550/arXiv.1412.6980.

Kondratev, A., Salminen, K., Rantala, J., Salpavaara, T., Verho, J., Surakka, V., Lekkala, J., Vehkaoja, A., and Müller, P. (2022). A comparison of online methods for change point detection in ion-mobility spectrometry data. *Array*, 14:100151. DOI: 10.1016/j.array.2022.100151.

Labiod, L. and Nadif, M. (2011). Co-clustering under nonnegative matrix tri-factorization. In *International Conference on Neural Information Processing*, pages 709–717. Springer. DOI: 10.1007/978-3-642-24958-7_82.

Lomet, A. (2012). *Sélection De Modèle Pour La Classification Croisée De Données Continues*. PhD thesis, Compiègne. http://www.theses.fr/2012COMP2041.

Marchello, G., Corneli, M., and Bouveyron, C. (2024). A deep dynamic latent block model for co-clustering of zero-inflated data matrices. *Journal of Computational and Graphical Statistics*, 1–16. DOI: 10.1080/10618600.2024.2319162.

Marchello, G., Fresse, A., Corneli, M., and Bouveyron, C. (2022). Co-clustering of evolving count matrices with the dynamic latent block model: Application to pharmacovigilance. *Statistics and Computing*, 32(3):1–22. DOI: 10.1007/s11222-022-10098-y.

Matias, C. and Miele, V. (2017). Statistical clustering of temporal networks through a dynamic stochastic block model. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(4):1119–1141. DOI: /10.1111/rssb.12200.

Nowicki, K. and Snijders, T. A. B. (2001). Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087. DOI: 10.1198/016214501753208735.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in **pytorch**. In *NIPS 2017 Autodiff Workshop*. https://openreview.net/forum?id=BJJsrmfCZ.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, ISSN: 01621459, http://www.jstor.org/stable/2284239.

Robert, V., Vasseur, Y., and Brault, V. (2021). Comparing high-dimensional partitions with the Co-clustering Adjusted Rand Index. *Journal of Classification*, 38:158–186, DOI: 10.1007/s00357-020-09379-w, https://hal.inria.fr/hal-01524832.

Selosse, M., Jacques, J., and Biernacki, C. (2020). Model-based co-clustering for mixed type data. *Computational Statistics & Data Analysis*, 144:106866. DOI: 10.1016/j.csda.2019.106866.

Yang, T., Chi, Y., Zhu, S., Gong, Y., and Jin, R. (2011). Detecting communities and their evolutions in dynamic social networks—a Bayesian approach. *Machine learning*, 82(2):157–189. DOI: 10.1007/s10994-010-5214-7.

A. Appendix

A.1. Update of the intensity parameter

Here our goal is to derive the online update of the zero-inflated Poisson intensity parameter, Λ. The variational distribution $q(A, Z, W)$ is kept fixed, while the lower bound is maximized with respect to Λ at every time instant t, to obtain its update, $\hat{\Lambda}$. To find the optimal update we compute the derivative of the lower bound $\mathcal{L}(q, \theta)$ in Eq. (11) with respect to Λ and set it equal to zero, as follows

$$\begin{aligned}
&\frac{\partial \log \mathcal{L}(q,\theta)}{\partial \Lambda_{q\ell}} = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{u=1}^{t}(1-\delta_{ij}(u))\left[\frac{\tau_{iq}(u)\eta_{j\ell}(u)X_{ij}(u)}{\Lambda_{q\ell}} - \tau_{iq}(u)\eta_{j\ell}(u)\right] = 0 \\
&\Leftrightarrow \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{u=1}^{t}(1-\delta_{ij}(u))\Big[\tau_{iq}(u)\eta_{j\ell}(u)X_{ij}(u) - \tau_{iq}(u)\eta_{j\ell}(u)\Lambda_{q\ell}\Big] = 0 \\
&\Leftrightarrow \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{u=1}^{t}(1-\delta_{ij}(u))\tau_{iq}(u)\eta_{j\ell}(u)\Lambda_{q\ell} = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{u=1}^{t}\tau_{iq}(u)\eta_{j\ell}(u)\Big[X_{ij}(u) - X_{ij}(u)\delta_{ij}(u)\Big] \\
&\Rightarrow \hat{\Lambda}_{q\ell} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{u=1}^{t}\tau_{iq}(u)\eta_{j\ell}(u)(X_{ij}(u)-\delta_{ij}(u)X_{ij}(u))}{\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{u=1}^{t}\tau_{iq}(u)\eta_{j\ell}(u)(1-\delta_{ij}(u))}. \qquad (15)
\end{aligned}$$

Although the update of $\hat{\Lambda}_{q\ell}$ sums over all the past observations $(X_{ij}(1), \ldots, X_{ij}(t))$, we can develop Eq. (15) as follows

$$\begin{aligned}
&\hat{\Lambda}_{q\ell} = \\
&\left[\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{u=1}^{(t-1)}\tau_{iq}(u)\eta_{j\ell}(u)(X_{ij}(u)-\delta_{ij}(u)X_{ij}(u)) + \sum_{i=1}^{N}\sum_{j=1}^{M}\tau_{iq}(t)\eta_{j\ell}(t)(X_{ij}(t)-\delta_{ij}(t)X_{ij}(t))\right] \\
&\Bigg/\left[\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{u=1}^{(t-1)}\tau_{iq}(u)\eta_{j\ell}(u)(1-\delta_{ij}(u)) + \sum_{i=1}^{N}\sum_{j=1}^{M}\tau_{iq}(t)\eta_{j\ell}(t)(1-\delta_{ij}(t))\right] \\
&= \frac{N_{q\ell}^{old} + N_{q\ell}^{(t)}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}} = \frac{N_{q\ell}^{old}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}} + \frac{N_{q\ell}^{(t)}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}}. \qquad (16)
\end{aligned}$$

By splitting Λ in two different parts, we can distinguish between a part known at time $t-1$, namely $N_{q\ell}^{old}$ and $D_{q\ell}^{old}$, and the current updates at time t, $N_{q\ell}^{(t)}$ and $D_{q\ell}^{(t)}$. Then, we divide and multiply the first term for $D_{q\ell}^{old}$, such that, denoting $\hat{\Lambda}_{q\ell}^{old} = N_{q\ell}^{old}/D_{q\ell}^{old}$, we obtain the final online update

$$\hat{\Lambda}_{q\ell} = \hat{\Lambda}_{q\ell}^{old} \cdot \frac{D_{q\ell}^{old}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}} + \frac{N_{q\ell}^{(t)}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}}. \qquad (17)$$

Hence, when a new observation comes Λ can be updated thanks to Eq.(17), along with the update of $D_{q\ell}^{old}$.

Algorithm 1 VEM-SGD Algorithm for Stream Zip-dLBM

Require: $X, \hat{Q}, \hat{L}, Q_{max}, L_{max}, max.iter, G_d(t)$.
Initialization of $\tau(t=0)$ and $\eta(t=0)$: sampling from $\mathcal{M}(\alpha(t=0))$ and $\mathcal{M}(\beta(t=0))$, respectively;
Initialization of $\delta(t=0)$: matrix of 1, then setting $\delta(t=0)=0$ when $X>0$;
while New observations $X(t)$ come: **do**
Initialization of $\alpha(t), \beta(t), \pi(t), \Lambda$ with LBM; % *with* $\hat{Q}$*, and* $\hat{L}$
▶ Add $Q_{max} - \hat{Q}$ columns of zeros to $\alpha(t)$;
▶ Add $L_{max} - \hat{L}$ columns of zeros to $\beta(t)$;
▶ Add $Q_{max} - \hat{Q}$ rows and $L_{max} - \hat{L}$ columns of zeros to Λ;
for it = 1 to $max.iter$ **do**
VE-Step:
for p = 1 to Fixed.Point **do**
alternatively update $\delta(t)$, $\tau(t)$, $\eta(t)$; % *fix point eqs*
end for
M-Step:
Update $\theta = (\Lambda, \pi(t), \alpha(t), \beta(t))$, $\hat{\Lambda}_{q\ell} = \hat{\Lambda}_{q\ell}^{old} \cdot \dfrac{D_{q\ell}^{old}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}} + \dfrac{N_{q\ell}^{(t)}}{D_{q\ell}^{old} + D_{q\ell}^{(t)}}$.
Update $\alpha(t), \beta(t), \pi(t)$ %*LSTM on the moving window* $t \in G_d(t)$
end for
Discard all the observation before $G_d(t)$.
end while

α			
cp	16	66	103
β			
cp	25	62	112
π			
cp	16	66	103

$$\Lambda = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ 7 & 3 \end{bmatrix}$$

Figure 5: Simulated time instants for change points of α, β and π and simulated values of Λ.

A.2. Pseudocode of the inference algorithm

The algorithm employs fixed-point equations to iteratively update the variational parameters ($\delta(t)$, $\tau(t)$, and $\eta(t)$) because of their interdependence. These iterations ensure convergence within the VE-Step for each new observation incoming. In our simulation studies, we found that setting the number of iterations (Fixed.Point) to 3 yielded good results.

A.3. Details on the simulated experiment

The simulated change points for the parameters α, β and π, and the simulated value of the intensity parameter Λ are shown in Figure 5.

A.4. Algorithmic considerations

In this section, we provide detailed technical specifications of the neural networks used in the M-Step of the inference algorithm. From $t = 1$ to $t = G_d(t)$ we employed fully connected neural networks with two hidden layers, consisting of 100 and 50 neurons, respectively. The choice of two hidden layers allows for capturing complex patterns and relationships within the data. When $t > G_d(t)$ LSTM neural networks are employed, as described in Section 3.3. The output of the LSTM is then reshaped and passed through two fully connected layers. Sigmoid activation is then applied after the first linear layer to introduce non-linearity, while softmax activation is applied after the second linear layer to obtain the probability distribution over the output classes. We did not used mini-batch training, opting for a full-batch approach where the training dataset used in each iteration consists in the data within the moving window $G_d(t)$. Within the VEM algorithm, each iteration involved the optimization of the lower bound. The fully connected neural networks have a learning rate of $\gamma = 1e^{-5}$, while the LSTM networks have a learning rate of $\gamma = 1e^{-3}$.In the experiments, this update is implemented in PyTorch via automatic differentiation (Paszke et al. 2017) and relies on stochastic optimisation (ADAM, Kingma and Ba 2014). Once the neural nets are trained via back-propagation (SGD) they provide us with the current ML estimates of $\alpha(t)$, $\beta(t)$ and $\pi(t)$. All the experiments run on CPU on a MacBook Pro, 2020, with a processor of 2,3 GHz Quad-Core Intel Core i7 and 16 GB of RAM.

Affiliation:

Giulia Marchello
Inria PreMeDICaL team, Idesp
Université de Montpellier
860 Rue de St - Priest, 34090 Montpellier
E-mail: giulia.marchello@inria.fr
URL: https://giuliamar95.github.io/giuliamar95/

Alexandre Destere
Université Côte d'Azur Medical Centre
Department of Clinical Pharmacology
Maasai team, Nice, France
E-mail: destere.a@chu-nice.fr

Marco Corneli
Université Côte d'Azur
Laboratoire CEPAM
Maasai team, Nice, France
E-mail: marco.corneli@inria.fr
URL: https://math.univ-cotedazur.fr/~mcorneli/

Charles Bouveyron
Université Côte d'Azur
Inria, CNRS, Laboratoire J.A.Dieudonné
Maasai team, Nice, France
E-mail:charles.bouveyron@inria.fr
URL: https://math.univ-cotedazur.fr/~cbouveyr/

Journal of Data Science, Statistics, and Visualisation https://jdssv.org/
published by the International Association for Statistical Computing
http://iasc-isi.org/

ISSN 2773-0689
August 2024, Volume IV, Issue IV *Submitted:* 2024-04-11
doi:10.52933/jdssv.v4i4.106 *Accepted:* 2024-08-01