IMPERIAL

MEng Individual Project

Imperial College London

Department of Computing

Scalable Dynamic Degree-Corrected Poisson Matrix Factorisation

Author:
Adrian-Ionut Boroica

Supervisor:
Dr Francesco Sanna Passino

Second Marker:
Professor Nick Heard

June 19, 2024

Abstract

In network science, entities such as users, servers, and machines are viewed as nodes, while the connections or interactions between them are referred to as links. Graph link prediction is crucial for understanding the reasons behind link formation and network behaviour. In the context of cyber-security, for instance, link prediction plays a vital role in identifying malicious behaviour and preventing cyber-attacks. Given the constantly evolving nature of large-scale networks, with nodes constantly joining, going offline, or becoming intermittently active, it is essential to incorporate node dynamics for accurate link prediction. The Poisson Matrix Factorization (PMF) model has proven effective for link prediction in extensive networks. The model assumes that every node has a latent position in a low-dimensional space, which is a mathematical embedding where each dimension captures underlying features of the nodes, and the probability of a link between two nodes depends only on their representation. However, this static representation fails to account for temporal changes, limiting its application to evolving networks.

In this report, we propose an extension of the PMF model that incorporates temporal node dynamics into its structure. We achieve this by assuming a fixed latent-space representation of the nodes, static over time, and correcting it with time-dependent scalar degree correction parameters. This allows the extended PMF model to capture the evolving nature of real-world networks. We formulate the model as a Bayesian hierarchical model and propose a scalable variational inference algorithm to fit it. Additionally, we provide a Python implementation of the model. To evaluate its performance, we apply the model to a randomly generated network using a stochastic block model and a real-world network flow dataset.

Acknowledgements

I am incredibly thankful to my supervisor, Dr. Francesco Sanna Passino, for his constant support, eagerness to help, and endless patience. His expert guidance and feedback have greatly improved my work and contributed immensely to my academic growth.

I also want to thank my family and friends for their unwavering support and encouragement. Their belief in me and constant reassurance have been a source of strength. Their emotional support, patience, and understanding have been essential in overcoming the challenges of my degree. I am deeply grateful for their love, which has played a crucial role in reaching this milestone.

Contents

Chapter 1

Introduction

In the dynamic landscape of modern networks, interactions occur in various forms, such as digital communications, in-person meetings, and business collaborations. These interactions form large, complex networks that encapsulate vast amounts of information about the entities involved, information that can be leveraged to anticipate future behaviours. The task of forecasting future connections within such extensive networks is known as link prediction.

The development of link prediction models is a vital task with broad implications across many different areas. Particularly in the realms of social media, biological networks, and cybersecurity, these models offer a powerful way of understanding and anticipating the evolution of complex systems. By accurately forecasting potential connections, link prediction models not only enhance the functionality of recommender systems and deepen our understanding of biological interactions, but also emerge as important tools in intrusion detection in cybersecurity.

In cybersecurity, these models take on a crucial role, identifying potential vulnerabilities and preventing malicious attacks by detecting anomalous patterns in network traffic. By predicting unusual links or changes in network behaviour, the model serves as an early warning system, flagging potential security breaches before they escalate.

Moreover, integrating anomaly detection capabilities enhances the model's effectiveness, allowing it to not just predict links, but also to discern irregularities within the network that could signify security threats. This dual functionality makes the model a valuable asset in network security, where anticipating and mitigating risks is as crucial as understanding the normal flow of network interactions.

In many real-world networks evolving over time, nodes tend to have similar connectivity patterns between different time snapshots, but their activity levels tend to differ (for example, some nodes might be more active during certain periods). In this report, our main research objective is to propose a statistical model to address this feature of real-world networks.

1.1 Project Motivation

Large networks are often dynamic, with new nodes joining, some nodes going offline, and others being intermittently active. When modelling real-world networks, it is essential to account for these temporal dynamics, as they can significantly impact the network structure and the interaction between nodes.

Let us consider the example of social media platforms (such as Twitter, Facebook, or Instagram). Here, the nodes represent users, and the links represent interactions between users, such as likes, comments, retweets, shares, or direct messages.

Users tend to have a core set of connections with whom they interact regularly. For example, close friends or family members frequently engage with each other's posts, creating a stable network of connections that persists over time. Furthermore, many users belong to communities (e.g. based on interests, locations, or affiliations) that influence their interactions with other users. These communities are relatively stable, though they may grow or shrink slightly over time.

But users' activity levels vary significantly with time due to factors such as daily routines, seasons, or specific events. For example, a user might be very active during the evenings or weekends, or during significant events like holidays, sports events, or elections. Moreover, certain events can cause spikes in activity. For example, a major news event or a viral trend can lead to a sudden increase in posts, shares, and comments, involving users who may otherwise be less active.

This temporal aspect of real-world networks is often overlooked in standard link prediction models, which assume that the network structure is static over time. Consequently, these models fail to capture the dynamic nature of networks where nodes and edges change. This omission leads to inaccurate predictions and an incomplete understanding of network's evolution.

For example, latent position models [Hoff et al., 2002] assign each node a latent position in a low-dimensional space, which is a mathematical embedding where each dimension captures underlying features of the nodes. In these models, the probability of a link between two nodes is a function of their latent positions, and is independent of any other links in the network. However, this representation does not change over time, which can be unrealistic in many scenarios.

One example of a latent position model is Poisson matrix factorization [Gopalan et al., 2015], abbreviated as PMF. The PMF model assumes that the number of links between two nodes follows a Poisson distribution, where the rate parameter is determined by the inner product of the nodes' latent positions. While PMF has been widely used in various applications, it does not account for temporal dynamics in the network structure.

Latent position models have been extended to a dynamic setting where the latent position $\mathbf{x}_{i,t} \in \mathbb{R}^d$ is expressed as a noisy function of values of the latent position at previous time points in numerous studies, such as Charlin et al. [2015], Kim et al. [2018] or Sewell and Chen [2015]. However, these studies have not addressed a frequently observed phenomenon in actual networks: the node varies its *activity level* over time, but not the type of connections which are formed.

This hints at a particular *degree-corrected form* of the latent positions: $\mathbf{x}_{i,t} = \rho_{i,t}\boldsymbol{\mu}_i$, where $\rho_{i,t} \in \mathbb{R}_+$ is a *time dependent* correction factor for the node activity, and $\boldsymbol{\mu}_i \in \mathcal{X}$ is a *time-invariant latent position.* Here, the correction factor allows to node to increase or decrease its activity level over time, while the latent position influences the type of connections that are formed. This approach would be valuable in fields where it is important to identify nodes exhibiting unusual changes in behaviour: for example, in cyber-security, it is not suspicious if a node increases or decreases its activity, but it is suspicious if it changes the type of connections it forms. Mathematically, changes over time in $\rho_{i,t}$ are allowed, while changes in $\boldsymbol{\mu}_i$ are flagged.

Building on this idea, we propose a dynamic extension of the PMF model that captures the temporal evolution of network structures. We do so by assuming a fixed latent-space representation of the nodes, static over time, corrected by scalar degree correction parameters, inspired by the celebrated degree-corrected stochastic block model [Karrer and Newman, 2011]. Furthermore, we propose scalable Bayesian inference procedure based on a variational approximation of the posterior distribution of the model parameters.

1.2 Related work

Link prediction can generally be categorized into static link prediction and dynamic link prediction. Static link prediction focuses on filling missing entries in an incomplete network graph [Clauset et al., 2008], which is widely used in fields like biological networks and social science research [Zhou et al., 2009]. On the other hand, dynamic link prediction, also known as temporal link prediction, aims to predict the emergence of new links over time in evolving networks [Gao et al., 2011]. Additionally, supervised learning models have been developed for link prediction in bipartite graphs. For instance, Benchettara et al. [2010] created a supervised model based on topological attributes of observed nodes in the graph (such as degree, common neighbours, and Jaccard's coefficient), using precision as the evaluation metric.

For improving the accuracy and efficiency of temporal link prediction, statisticians have proposed various models. Hoff et al. [2002] introduced *latent position models*, where the probability of a link between two nodes depends only on the latent positions of the two nodes in an unobserved low-dimensional space (and is independent on any other links in the network). Possible examples of such models are distance models and projection models. Distance models compute the probability of observing links as a function of the distance between nodes, while projection models calculate link probability based on a linear projection of the interacting nodes. A widely used latent position model is the Poisson matrix factorization model (PMF) developed by Gopalan et al. [2015]. The model is based on the assumption that the number of links between two nodes follows a Poisson distribution, where the rate parameter is determined by the inner product of the nodes' latent positions. To adapt PMF for cybersecurity applications, Sanna Passino et al. [2022] incorporated indicator functions to include categorical covariates and also modified it to account for seasonal dynamics.

The PMF model has been extensively adapted and modified for various purposes. While the original PMF model assumes static and time-invariant latent features, Charlin et al. [2015] developed a time-dependent latent position PMF model to improve prediction accuracy for changes in customer interest and population shifts. For analysing sequential count vectors, Gong and Huang [2017] proposed the Deep Dynamic Poisson Factor Analysis (DDPFA) model, which uses mean field variational inference and a deep structure to capture long-term dependencies between count vectors. Additionally, Hosseini et al. [2017] adapted the PMF model for online shop recommendation systems. In their model, the likelihood of a customer liking a product is determined by two independent components: the intrinsic part, which reflects the compatibility between the customer and the product, and the extrinsic part, which indicates the customer's tendency to engage with items based on past interactions.

Besides PMF-based models, there are other latent position models that can be adapted for dynamic link prediction, such as non-negative matrix factorization models [Chen et al., 2017] and random dot product graphs [Young and Scheinerman, 2007; Athreya et al., 2018]. For instance, Sanna Passino et al. [2021] enhanced link prediction performance on temporal dynamic networks by applying random dot product graph techniques with a dynamic framework.

1.3 Structure of the report

The rest of the report is organized as follows. Chapter 2 provides the foundational knowledge necessary for understanding the main algorithm discussed in the report. Chapter 3 presents the model and the variational inference algorithm for fitting it, including all the necessary proofs. Chapter 4 describes the simulation study, where we evaluate the model's performance on synthetic data. Chapter 5 applies the model to a real-world dataset and discusses the results. Chapter 6 concludes the report and outlines potential future research directions.

The corresponding code for the model implementation, simulation study, and real-data analysis can be found on GitHub at: `https://github.com/adiboroica/sddc-pmf`.

Chapter 2

Background

2.1 Graphs

A **graph** (or **network**) [Newman, 2018] $G = (V, E)$ is a structure comprised of a set of nodes V and a set of edges $E \subseteq V \times V$. For each graph, we can construct an **adjacency matrix** $\mathbf{A} \in \{0,1\}^{|V_1|\times|V_2|}$, which records the presence of edges:

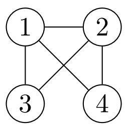
$$A_{ij} \overset{\text{def.}}{=} \mathbb{1}_E\{(i,j)\},$$

where the **indicator function** of a set B is defined as:

$$\mathbb{1}_B(x) \overset{\text{def.}}{=} \begin{cases} 1 & \text{if } x \in B, \\ 0 & \text{otherwise.} \end{cases}.$$

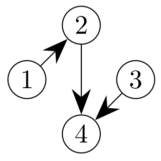
An **undirected graph** does not take into account the direction of edges: the edge (i,j) is the same as the edge (j,i) (for all $i,j \in V$). Note that in this case the adjacency matrix is symmetric.
On the other hand, a **directed graph** takes into the account the direction of edges: usually, $(i,j) \in E$ does not imply $(j,i) \in E$, and the adjacency matrix is not necessarily symmetric.

In a **bipartite graph**, the vertices can be divided into two *disjoint* sets V_1 and V_2 such that $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, and every edge has a node in V_1 and a node in V_2 (sometimes they are called source and destination nodes). In this case, the adjacency matrix is a $|V_1| \times |V_2|$ matrix, since there are no edges between nodes in the same set.



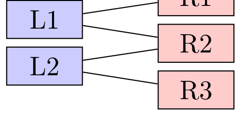
Undirected graph G_1

$$A_1 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$



Directed graph G_2

$$A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



Bipartite graph G_3

$$A_3 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Figure 2.1: Examples of directed, undirected and bipartite graphs with corresponding adjacency matrices

The **degree** of a node in a graph represents the number of edges connected to the node. In the case of an undirected graph, this simply counts the total number of connections to the node. In a directed graph, the degree can be split into two types: the **in-degree** and the **out-degree**. The **in-degree** of a node specifically refers to the number of incoming edges, that is, the edges that point to the node from other nodes in the graph. The **out-degree** of a node specifically refers to the number of outgoing edges from that node, that is, the edges start from that node and point to other nodes in that graph.

2.2 Latent Position Model

A **discrete dynamic graph model** consists of:

1. A discrete sequence of graphs $\mathbb{G}_1, \cdots, \mathbb{G}_T$ (where $T \in \mathbb{N}_{>0}$) is observed.
2. Each graph $\mathbb{G}_t$ is determined by a set of nodes V (where V is the set of *all possible nodes*) and by a set of edges $E_t \subseteq V \times V$. We have that $(i, j) \in E_t$ (where $i, j \in V$) if a connection between the two nodes is observed at time t.
3. In some applications, we care only about the existence of a connection, so this sequence of graphs can be analysed by considering the adjacency matrices $A_1, \cdots, A_T$. Here, $A_{tij} \overset{\text{def.}}{=} \mathbb{1}_{E_t}\{(i, j)\}$ tells us if there is an edge between nodes i and j at time t.

Consider an example where we use this model to represent the traffic in an airport network over a week:

- Assuming that we are monitoring the network on a daily basis, we have $T = 7$.
- The nodes in this model symbolize the various airports within the network.
- Edges between any two nodes (airports) are defined by the total number of flights that operate between these airports each day. The weight of an edge is thus a quantitative measure of the flight traffic between the corresponding airports.

This model is particularly useful for analysing and optimizing both flight schedules and overall airport connectivity. The variable edge weights, which change daily, are reflective of the frequency of flights on each route. This dynamic aspect of the model is crucial for effective management of airport traffic and strategic planning of new flight routes. Figure 2.2 shows a possible realization of this model (weights are omitted for simplicity).

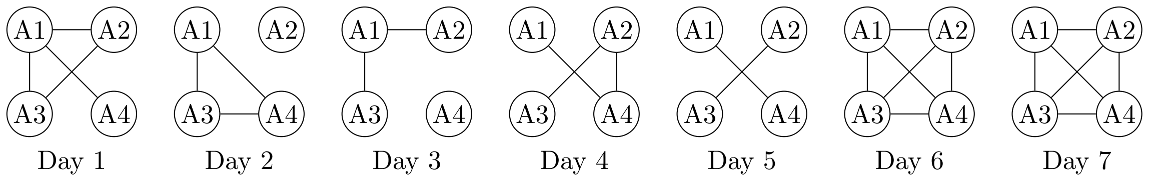


Figure 2.2: Example of airport traffic of a network consisting of 4 airports during a week

By adapting the **latent position model** [Hoff et al., 2002], we can derive a specific type of discrete dynamic graph model, based on the following assumptions:

1. Each node has a **latent lower dimensional representation** $\mathbf{x}_{i,t} \in \mathcal{X} \subseteq \mathbb{R}^d$. This is a set of unobservable variables that influence the network's structure, which have a lower dimensionality than the previous state or feature space.

   These latent representations capture the essential characteristics and relationships of the nodes in a more compact form, facilitating efficient analysis and modelling of the network. This reduction in dimensionality not only lowers computational complexity, making it feasible to process and analyse large networks, but also aids in visualizing complex data structures. For instance, in social network analysis, latent representations can reveal hidden communities or influential nodes.

2. The probability of each edge can be described as a function of these variables:

$$A_{i,j,t} \sim \text{Bernoulli}\{\kappa(\mathbf{x}_{i,t}, \mathbf{x}_{j,t})\},$$

where $\kappa\colon \mathcal{X} \times \mathcal{X} \to [0, 1]$ is called the kernel function.

One common kernel function used in literature is the **binary PMF link** (see, for example, Caron and Fox [2017] or Sanna Passino et al. [2022]):

$$\kappa(\mathbf{x}, \mathbf{x}^*) = 1 - \exp(-\mathbf{x}^\top \mathbf{x}^*),$$

which has computationally convenient properties related to sparsity and scalability to large graphs. In this case we must assume that the latent features are non-negative.

Let us see how we could add these assumptions to our previous example:

- We associate each node (airport) with a latent position $\mathbf{x}_{i,t}$ in a lower-dimensional space $\mathcal{X} \subseteq \mathbb{R}^d$. This position could represent factors like geographic location, airport capacity, or other operational characteristics that are not directly observable but influence the network's dynamics.
- In this case, we may use the *exponential euclidean kernel*: $\kappa(\mathbf{x}_{i,t}, \mathbf{x}_{j,t}) = \exp(-\alpha|\mathbf{x}_{i,t} - \mathbf{x}_{j,t}|)$. Here, $|\mathbf{x}_{i,t} - \mathbf{x}_{j,t}|$ represents the Euclidean distance between the latent positions of the two airports, and α is a positive parameter that modulates the influence of this distance on the connection probability.

This enriched model provides a more nuanced view of the airport network. For instance, it could help in understanding how changes in certain unobservable factors at an airport (like security measures, management efficiency, etc.) influence its connectivity with other airports over the course of time. Moreover, it could be used to predict the formation of new routes or the closure of the current ones.

2.3 Matrix Factorization Models for temporal link prediction

Matrix factorization models for link prediction are a class of algorithms that aim to predict the presence or strength of links in networks, such as social networks, user-item interaction networks, or any type of graph structure. These models are particularly effective in collaborative filtering tasks, such as in recommender systems where the goal is to suggest new products or services to users based on past interactions.

The main idea behind matrix factorization models is to represent the nodes in the network as vectors in a latent space, where the similarity between nodes is measured by the inner product of their corresponding vectors. The model learns these latent representations by factorizing the adjacency matrix of the network into two lower-dimensional matrices, which are then used to predict the missing or future links in the network.

We are going to present two such models: PMF (on which our proposed model is based on) and RDPG (which is going to be used as a baseline in our experiments).

2.3.1 Poisson Matrix Factorization (PMF)

Poisson Matrix Factorization (PMF) [Gopalan et al., 2015] is a probabilistic matrix-factorisation model, with applications in various fields such as recommendation systems, image processing, and biological data analysis.

Its original use was to produce high-quality recommendations to customers. Based on past interactions, the system makes predictions about the possible ratings a specific user can give to a particular product. We define the model in this context as follows:

Let V_1 be the set of all N_1 users, with $i \in \{1, \cdots, N_1\}$ denoting the user index. Similarly, let V_2 be the set of all N_2 products, with $j \in \{1, \cdots, N_2\}$ denoting the product index.

We may have a lot of information about the users and products that can be used to make predictions. However, not all of this information is necessarily relevant, and processing high-dimensional data can be computationally expensive.

Therefore, it is often advantageous to derive a lower-dimensional representation of the data that captures the essential characteristics and relationships of the users and products. This representation is referred to as **latent position features**, denoted as $\mathbf{x}_i \in \mathbb{R}_+^d$ for user i and $\mathbf{y}_j \in \mathbb{R}_+^d$ for product j. These latent features are unobservable variables that influence the interactions between users and products.

Notice that the number of latent features for a user must be the same as the number of latent features for a product, denoted as d. We use $r \in \{1, \cdots, d\}$ to represent the latent feature index.

The ratings are represented by a non-negative matrix $\mathbf{N} \in \mathbb{N}^{|V_1| \times |V_2|}$, where N_{ij} represents the rating user i gives to product j. Since we are dealing with integer-valued data, the Poisson distribution is a natural choice for modelling the ratings:

$$N_{ij} \sim \text{Poisson}(\mathbf{x}_i^T \mathbf{y}_j), \text{ where } \mathbf{x}_i^T \mathbf{y}_j = \sum_{r=1}^{d} x_{ir} y_{jr}.$$

To predict the future rating a specific user might give to a particular product, the system first estimates latent positions from the past interactions between users and products, and then uses these estimates for making the predictions.

A graphical representation of the PMF model is shown in Figure 2.3.

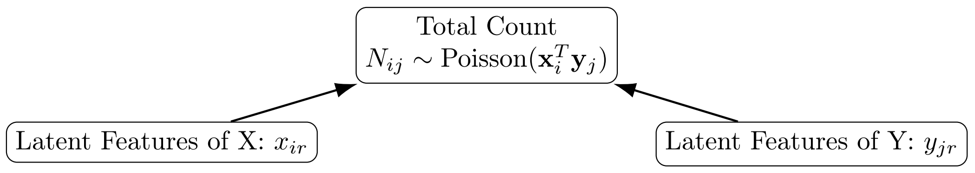

Total Count
$N_{ij} \sim \text{Poisson}(\mathbf{x}_i^T \mathbf{y}_j)$
Latent Features of X: x_{ir}
Latent Features of Y: y_{jr}

Figure 2.3: Diagram of the PMF Model. Here, the total count represents the ratings from our example.

In the **Binary PMF** model, the edges are modelled by an adjacency matrix $\mathbf{A} \in \{0,1\}^{|V_1| \times |V_2|}$, since we are dealing with a bipartite graph. For example, in cyber-security link prediction tasks, we care only about the existence of an interaction between the nodes. Notice that the ranges between this model and the PMF model are mismatched: the Poisson distribution can take any non-negative value, while our edges can only take 0 or 1. In order to adapt the PMF model to the specified range, we apply an indicator function as follows:

$$A_{ij} = \mathbb{1}_{\mathbb{N}_+}(N_{ij}).$$

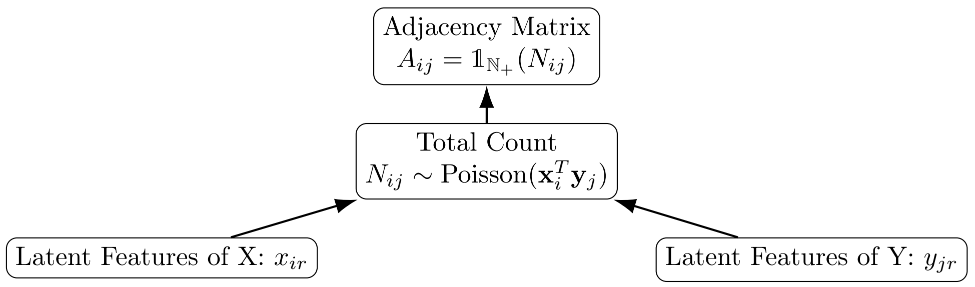

Adjacency Matrix
$A_{ij} = \mathbb{1}_{\mathbb{N}_+}(N_{ij})$
Total Count
$N_{ij} \sim \text{Poisson}(\mathbf{x}_i^T \mathbf{y}_j)$
Latent Features of X: x_{ir}
Latent Features of Y: y_{jr}

Figure 2.4: Diagram of the Binary PMF Model.

A graphical representation of the Binary PMF model is shown in Figure 2.4.

In the truncated model, the entry A_{ij} follows a Bernoulli distribution with $\mathbb{P}[A_{ij} = 0] = \mathbb{P}[N_{ij} = 0]$:

$$A_{ij} \sim \text{Bernoulli}(1 - \exp[-\mathbf{x}_i^T \mathbf{y}_j]).$$

Under a Bayesian framework, Gopalan et al. [2015] applied Gamma hierarchical priors on these latent features to enhance the relationships between the parameters, such that:

$$\begin{aligned}
x_{ir} &\sim \text{Gamma}(a^{(x)}, \zeta_i^{(x)}) \ , \ \zeta_i^{(x)} \sim \text{Gamma}(b^{(x)}, c^{(x)}), \\
y_{jr} &\sim \text{Gamma}(a^{(y)}, \zeta_j^{(y)}) \ , \ \zeta_j^{(y)} \sim \text{Gamma}(b^{(y)}, c^{(y)}), \\
i &= 1, \cdots, N_1 \ , \ j = 1, \cdots, N_2 \ , \ r = 1, \cdots, d.
\end{aligned}$$

where $a^{(x)}, a^{(y)}, b^{(x)}, b^{(y)}, c^{(x)}, c^{(y)} \in \mathbb{R}_+$ are deterministic non-negative constants.

Dynamical Extensions

As we can see, the PMF models the latent features statically over time, which is unrealistic (consider the case of user preferences, which clearly evolve over time). In numerous works in literature, the PMF model has been extended to dynamic setting, such as the dPF (Dynamic Poisson Matrix) model [Charlin et al., 2015], which was used to model time series of clicks in recommender systems.

Shi [2022] and Li [2023] have introduced *node activation* parameters Q_{it} and Q^*_{jt} with *node activation rates* θ_i and θ^*_j in order to represent time-varying activities of source and destination nodes. Here, the value of Q_{it} indicates if the source node i is active (1) or inactive (0) at time t, while the value of Q^*_{jt} indicates if the destination node j is active (1) or inactive (0) at time t. Moreover, the node activation rates show how frequently a certain node is active.

Since a Bayesian hierarchical model approach makes the lower-level parameters less sensitive to noise by combining information from multiple observed networks, Beta priors are placed on the node activation rates.

Overall, the full model is expressed as:

$$\begin{aligned}
A_{ijt} &= \mathbb{1}_{\mathbb{N}_+}(N_{ijt}), \\
N_{ijt} &= Q_{it} Q^*_{jt} \cdot \text{Poisson}(\mathbf{x}_i \mathbf{y}_j), \\
Q_{it} &\sim \text{Bernoulli}(\theta_i) \ , \ \theta_i \sim \text{Beta}(\alpha^{(x)}, \beta^{(x)}), \\
Q^*_{jt} &\sim \text{Bernoulli}(\theta^*_j) \ , \ \theta^*_j \sim \text{Beta}(\alpha^{(y)}, \beta^{(y)}), \\
x_{ir} &\sim \text{Gamma}(a^{(x)}, \zeta_i^{(x)}) \ , \ \zeta_i^{(x)} \sim \text{Gamma}(b^{(x)}, c^{(x)}), \\
y_{jr} &\sim \text{Gamma}(a^{(y)}, \zeta_j^{(y)}) \ , \ \zeta_j^{(y)} \sim \text{Gamma}(b^{(y)}, c^{(y)}),
\end{aligned}$$

where $\alpha^{(x)}, \alpha^{(y)}, \beta^{(x)}, \beta^{(y)}, a^{(x)}, a^{(y)}, b^{(x)}, b^{(y)}, c^{(x)}, c^{(y)} \in \mathbb{R}_+$ are deterministic non-negative constants and

Note that the conditional distribution of N_{ijt} can be written as:

$$\begin{aligned}
N_{ijt} &= Q_{it} Q^*_{jt} \cdot \text{Poisson}(\mathbf{x}_i \mathbf{y}_j) \\
&\sim \begin{cases} \text{Poisson}(\mathbf{x}_i \mathbf{y}_j) & \text{if } Q_{it} = Q^*_{jt} = 1, \\ \delta_0 & \text{if } Q_{it} = 0 \text{ or } Q^*_{jt} = 0. \end{cases}
\end{aligned}$$

If $Q_{it} = 0$ or $Q^*_{jt} = 0$ for some indexes i, j, t, then the corresponding N_{ijt} will be 0 with probability 1, which is the Dirac delta distribution at point 0 (i.e. δ_0).

Shi [2022] has used Markov Chain Monte Carlo in order to sample from the posterior distribution, while Li [2023] has used Variational Inference in order to approximate the true posterior.

2.3.2 Random Dot Product Graph (RDPG)

The **Random Dot Product Graph (RDPG)** model [Young and Scheinerman, 2007; Athreya et al., 2018] is another popular method used in cyber-security for predicting temporal links.

It constructs the edge features by equating the probability of connecting a source node i with a destination node j to the inner product of their respective latent features, x_i and y_j, represented as:

$$P(A_{ij} = 1) = x_i^T y_j.$$

Since this study focuses exclusively on bipartite graph link prediction, the following sections will be dedicated to discussing embedding techniques and RDPG strategies specifically related to bipartite graphs.

Embedding methods

The choice of the embedding method depends on whether the graph is directed or undirected.

For undirected graphs, where the adjacency matrix is symmetric, *Adjacency Spectral Embedding* (ASE) is a popular choice, while for directed graphs, *Adjacency Embedding of the Directed graph* (DASE) is used. We omit the presentation of ASE, as our project focuses on directed graphs.

Furthermore, for a sequence of adjacency matrices, *Multiple Adjacency Spectral Embedding* (MASE), as described by Arroyo et al. [2021], effectively identifies common spectral embedding positions and dynamic scaling matrices across all observed adjacency matrices.

Adjacency Embedding of the Directed graph (DASE)

For bipartite graphs, the adjacency matrix A is a rectangular $0, 1$ matrix of dimensions $M \times N$. We want to estimate the latent features X and Y of the source and destination nodes, respectively. After selecting the number of features d, we apply singular value decomposition (SVD) to A, which can be represented as:

$$A = \begin{bmatrix} U & U_\perp \end{bmatrix} \begin{bmatrix} D & 0 \\ 0 & D_\perp \end{bmatrix} \begin{bmatrix} V^T \\ V_\perp^T \end{bmatrix} = UDV^T + U_\perp D_\perp V_\perp^T,$$

where D is a $d \times d$ diagonal matrix with the largest d singular values in descending order, and U (dimensions $M \times d$) and V (dimensions $N \times d$) are the corresponding orthogonal left and right singular vectors, respectively. Then we estimate the latent features as:

$$\widehat{X} = UD^{1/2} \in \mathbb{R}^{M\times d} \text{ and } \widehat{Y} = VD^{1/2} \in \mathbb{R}^{N\times d}.$$

Intuitively, we keep the most important features of the adjacency matrix A (which are given by the largest singular values) and discard the rest. Mathematically, this is a low-rank approximation of A:

$$A \approx UDV^T = \widehat{X}\widehat{Y}^T.$$

Multiple Adjacency Spectral Embedding (MASE)

Now suppose that we have multiple adjacency matrices $A_1, \ldots, A_T$, and we want to estimate the time-variant latent features X_t and Y_t on each adjacency matrix A_t. After selecting the number of features d, we use DASE to estimate the time-variant latent features X_t and Y_t on each adjacency matrix A_t:

$$X_t = U_t D_t^{1/2} \in \mathbb{R}^{M\times d} \text{ and } Y_t = V_t D_t^{1/2} \in \mathbb{R}^{N\times d}.$$

In order to perform spectral embedding across all adjacency matrices, we construct joint matrices U and V as follows:

$$U = [U_1 \ldots U_T] \in \mathbb{R}^{M\times Td} \text{ and } V = [V_1 \ldots V_T] \in \mathbb{R}^{N\times Td},$$

and then apply SVD to U and V, resulting in:

$$U = U_X D_X V_X^T + U_{X,\perp} D_{X,\perp} V_{X,\perp}^T \text{ and } V = U_Y D_Y V_Y^T + U_{Y,\perp} D_{Y,\perp} V_{Y,\perp}^T,$$

where D_X and D_Y are $d \times d$ diagonal matrices containing the top d singular values in descending order, $U_X \in \mathbb{R}^{M\times d}$, $U_Y \in \mathbb{R}^{N\times d}$, $V_X \in \mathbb{R}^{M\times d}$, and $V_Y \in \mathbb{R}^{N\times d}$ include the corresponding orthogonal left and right singular vectors.
As outlined in Arroyo et al. [2021], the MASE latent features $\widehat{X}$ and $\widehat{Y}$ are given by U_X and U_Y, respectively. Furthermore, the time-variant scaling matrix R_t captures the interactions between nodes at time t, influenced by the overall structure derived from all matrices. This matrix is computed as follows:

$$R_t = U_X^T A_t U_Y \in \mathbb{R}^{d\times d}, \text{ for } 1 \leq t \leq T.$$

RDPG Models

Given the adjacency matrices $\mathbf{A}_1, \ldots, \mathbf{A}_T$, there are many models that can be used to predict the next adjacency matrix $\mathbf{A}_{T+1}$. Two of the most prominent models are the *Averages of Inner Products of Embeddings (AIP)* [Dunlavy et al., 2011] and the *Common Subspace Independent Edge (COSIE)* [Arroyo et al., 2021].

In such models, the link probability between nodes i and j at time $T+1$ is estimated as:

$$\widehat{P}(A_{ij,T+1}) = \widehat{\mathbb{E}}[A_{ij,T+1}],$$

where $\widehat{\mathbb{E}}[A_{T+1}]$ represents the predicted values of the future adjacency matrix $\mathbf{A}_{T+1}$.

These two models will be used to compare the performance of our proposed model in predicting the future adjacency matrix $\mathbf{A}_{T+1}$.

Averages of Inner Products of Embeddings (AIP)

The future matrix $\mathbf{A}_{T+1}$ is forecasted as a linear combination of the expected values of previously observed matrices $\mathbf{A}_1, \ldots, \mathbf{A}_T$. This is mathematically represented as:

$$\widehat{\mathbb{E}}[\mathbf{A}_{T+1}] = \sum_{t=1}^{T} \theta_t \cdot \widehat{\mathbb{E}}[\mathbf{A}_t] = \sum_{t=1}^{T} \theta_t \cdot \widehat{X}_t \widehat{Y}_t^T,$$

where $\{\theta_t\}$ are positive weights that sum to 1 and $\widehat{\mathbb{E}}[\mathbf{A}_t] = \widehat{X}_t \widehat{Y}_t^T$ is the low-rank approximation of $\mathbf{A}_t$ obtained from the DASE method.

Scheinerman and Tucker [2010] recommend using an equal weight for all matrices, i.e. $\theta_t = \frac{1}{T}$ for each t, in what is referred to as the unweighted AIP. This approach assumes that the adjacency matrices $\mathbf{A}_1, \ldots, \mathbf{A}_T$ are independently generated from the same Bernoulli distribution and do not account for temporal changes.

Common Subspace Independent Edge (COSIE)

This model aims to identify shared embedding positions across all dynamic adjacency matrices $\mathbf{A}_1, \ldots, \mathbf{A}_T$ using the MASE method, and then construct a scaling matrix that, together with the shared embedding positions, can be used to predict the next adjacency matrix $\mathbf{A}_{T+1}$.

Let $\widehat{X}$ and $\widehat{Y}$ be the shared embedding positions obtained from the MASE method, and $R_1, \ldots, R_T$ be the scaling matrices for each adjacency matrix. The prediction for $\mathbf{A}_{T+1}$ is estimated as:

$$\widehat{\mathbb{E}}[\mathbf{A}_{T+1}] = \widehat{X} \left(\frac{1}{T} \sum_{t=1}^{T} R_t \right) \widehat{Y}^T.$$

Here, the scaling matrix for $\mathbf{A}_{T+1}$ is derived by averaging the scaling matrices from $\mathbf{R}_1$ to $\mathbf{R}_T$.

2.4 Variational Inference (VI)

In order to tackle the disadvantages of the Markov Chain Monte Carlo (MCMC) methods (a review of them can be found in Appendix A), we are going to introduce Variational Inference (VI), as it is described in Blei et al. [2017]. VI is a computationally efficient alternative which transforms inference into an optimization problem, focusing on approximating complex distributions with simpler ones for scalability. It involves selecting a distribution family and identifying the closest member to the target distribution, usually using Kullback-Leibler (KL) Divergence. While VI is faster and more scalable, it sometimes trades off accuracy for efficiency.

Kullback-Leiber (KL) Divergence is a measure from the field of information theory, often used to quantify how one probability distribution p differs from a reference probability q. Intuitively, it measures the amount of information lost when approximating one distribution with another. Mathematically, it is defined as:

$$\mathrm{KL}(p \parallel q) \overset{\text{def.}}{=} \mathbb{E}_{X\sim p}\left\{\log \frac{p(X)}{q(X)}\right\}.$$

Here are some properties of this measure:

1. If there is a point such that $p(x) \neq 0$ and $q(x) = 0$, then the KL divergence is defined as infinity.
2. The KL divergence is *non-negative* and minimized when $q = p$.
3. It is often described as a measure of the 'distance' between two probability distributions, although it is not a true metric since it is *asymmetric*. Thus, it is important which distribution is used as the approximation and which as the reference.

In **variational inference**, we want to approximate a conditional density of *latent variables* $\mathbf{z} = z_{1:m}$ given *observed variables* $\mathbf{x} = x_{1:n}$. To do so, we specify a family $\mathcal{Q}$ of densities over the random variables, where each $q(\mathbf{z}) \in \mathcal{Q}$ is a possible approximation of the exact conditional. In order to find the best candidate, we pick the one closest in KL divergence to the exact conditional.

$$q^*(\mathbf{z}) \overset{\text{def.}}{=} \underset{q(\mathbf{z})\in\mathcal{Q}}{\arg\min}\, \mathrm{KL}(q(\mathbf{z}) \parallel p(\mathbf{z} \mid \mathbf{x})).$$

Now let us expand this KL divergence:

$$\begin{aligned}\mathrm{KL}(q(\mathbf{z}) \parallel p(\mathbf{z} \mid \mathbf{x})) &= \mathbb{E}_q\left[\log \frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x})}\right] = \mathbb{E}_q\left[\log q(\mathbf{z})\right] - \mathbb{E}_q\left[\log p(\mathbf{z} \mid \mathbf{x})\right] \\ &= \mathbb{E}_q\left[\log q(\mathbf{z})\right] - \mathbb{E}_q\left[\log p(\mathbf{z}, \mathbf{x})\right] + \log p(\mathbf{x}).\end{aligned}$$

The previous KL divergence is not computable since the *evidence* $p(\mathbf{x})$ is hard to compute. This motivates the introduction of the **Evidence Lower Bound (ELBO)**:

$$\mathrm{ELBO}(q \parallel p) \overset{\text{def.}}{=} \mathbb{E}_q\left[\log p(\mathbf{z}, \mathbf{x})\right] - \mathbb{E}_q\left[\log q(\mathbf{z})\right].$$

Notice that this is indeed a lower bound for the evidence since the KL divergence is non-negative.

The problem of minimizing the KL divergence is thus equivalent to maximizing the ELBO.

2.4.1 Mean-Field Variational Family

In order to simplify our problem of minimizing the ELBO, we can add more assumptions to our variational family: suppose that each latent variable is mutually independent and is controlled by a separate factor within the variational family. This is called a **mean-field variational family**, where each member can be written as:

$$q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j).$$

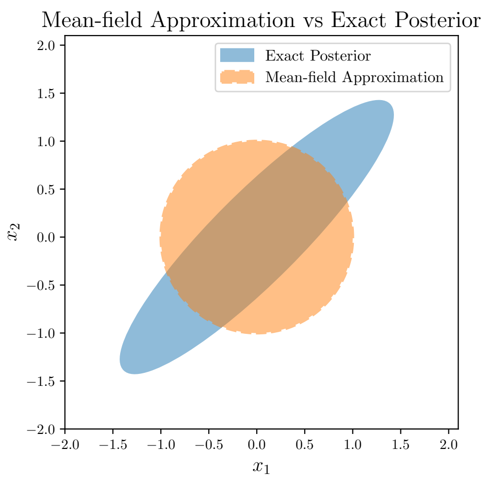


Figure 2.5: Comparison of the exact posterior and mean-field variational inference for a two-dimensional Gaussian. The mean-field approximation assumes variable independence, resulting in an underestimation of the true posterior variance.

Figure 2.5 shows the mean-field variational inference for a two-dimensional Gaussian.

As we see, the assumption that the latent variables are independent oversimplifies complex data by ignoring the dependencies between the latent variables. This can lead to inaccurate modelling and inference, as it fails to capture the true underlying relationships in intricate datasets.

But there is a computational advantage to this assumption. We are now going to present **CAVI (Coordinate Ascent Variational Inference)**, which is an algorithm that climbs the ELBO to a local optimum by iteratively optimizing each factor of the mean-field variational density.

The **complete conditional** of z_j (jth latent variable) is the conditional density of z_j given the observations and all the other latent variables, that is, $p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})$. If we fix the distributions $q_l(z_l), l \neq j$ of the other variational factors, then the optimal distribution of z_j is given by

$$\begin{aligned} q_j^*(z_j) &\propto \exp\{\mathbb{E}_{-j}[\log p(z_j \mid \mathbf{z}_{-j}, \mathbf{x})]\} \\ &\propto \exp\{\mathbb{E}_{-j}[\log p(z_j, \mathbf{z}_{-j}, \mathbf{x})]\}, \end{aligned}$$

where the expectation is considered with respect to the variational densities over $\mathbf{z}_{-j}$ (remember that these are currently fixed). Since all the latent variables are independent, the expectation on the right-hand side does not involve the jth variational factor, so this is a valid coordinate update.

Algorithm 1: Coordinate Ascent Variational Inference (CAVI)

```
Input: A model $p(\mathbf{x}, \mathbf{z})$, a data set $\mathbf{x}$
Output: A mean-field variational density $q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j)$.
Initialize: Mean-field Variational factors $q_j(z_j)$.
while the ELBO has not converged do
    for $j \in \{1, \ldots, m\}$ do
        Set $q_j(z_j) \propto \exp\{\mathbb{E}_{-j}[\log p(z_j|\mathbf{z}_{-j}, \mathbf{x})]\}$.
    end
    Compute $\mathrm{ELBO}(q \parallel p) = \mathbb{E}[\log p(\mathbf{z}, \mathbf{x})] - \mathbb{E}[\log q(\mathbf{z})]$.
end
return $q(\mathbf{z})$
```

The pseudocode for CAVI is presented in Algorithm 1.

Notice that the assumption that the latent features are independent is critical in order to assure the correctness of the algorithm. If the latent features are dependent, other optimization techniques must be used (for example, see Charlin et al. [2015]).

2.4.2 Advantages and Disadvantages

Variational Inference can be a better option than Markov Chain Monte Carlo in certain scenarios, primarily due to the following reasons:

- **Computational Efficiency**: VI often converges faster than MCMC. This is particularly beneficial in large-scale problems or when dealing with massive datasets, where MCMC can be prohibitively slow.
- **Deterministic Nature**: Unlike MCMC, which is inherently stochastic and involves randomness in the sampling process, VI is deterministic. This can lead to more predictable and stable behaviour in some applications.
- **Scalability**: VI scales better with large data and model sizes. This makes it suitable for modern machine learning problems, which often involve complex models and vast amounts of data.
- **Online and Streaming Data Adaptation**: VI can be more readily adapted to online and streaming data scenarios. The deterministic nature of VI allows it to update the approximations efficiently as new data arrives.
- **Well-suited for Optimization Frameworks**: VI's formulation as an optimization problem fits well into the existing frameworks used in machine learning, which are often optimization-centric.

However, it is important to note that VI also has its drawbacks. Sometimes, it might not capture the full complexity of the target distribution (e.g. when dealing with multimodal distributions) as accurately as MCMC can.

Moreover, VI gives us an *approximate distribution*, while MCMC guarantees samples that *asymptotically converge to the true posterior distribution* if run for a sufficient amount of time and under the right conditions.

The choice between VI and MCMC depends on the specific requirements of the problem at hand, including the trade-off between accuracy and computational efficiency.

2.5 Gamma Distribution

2.5.1 Gamma

The **Gamma** distribution is defined as:

$$\text{Gamma}\left(x\ ;\ \alpha, \beta\right) \overset{\text{def.}}{=} \mathbb{1}_{>0}(x) \times \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x},$$

where $\alpha \in \mathbb{R}_{>0}$ and $\beta \in \mathbb{R}_{>0}$ are the *shape* and *rate* of the distribution, and the Gamma function is given by:

$$\Gamma(z) \overset{\text{def.}}{=} \int_0^\infty t^{z-1} e^{-t}\, \mathrm{d}t.$$

The expectation of the Gamma distribution is given by:

$$\mathbb{E}\left[\text{Gamma}(\alpha, \beta)\right] = \frac{\alpha}{\beta}.$$

Moreover, it is well known that the expectation of the log-Gamma distribution can be directly calculated using the digamma function $\psi(z) \overset{\text{def.}}{=} \frac{\mathrm{d}}{\mathrm{d}z} \log \Gamma(z) = \frac{\Gamma'(z)}{\Gamma(z)}$:

$$\mathbb{E}\left[\log \text{Gamma}(\alpha, \beta)\right] = \psi(\alpha) - \log(\beta). \tag{2.1}$$

This formula provides a straightforward way to calculate the expected value without the need for numerical integration.

2.5.2 Truncated Gamma

Let us start by defining the *lower and upper incomplete gamma functions*:

$$\Gamma(\alpha, x) \overset{\text{def.}}{=} \int_x^\infty t^{\alpha-1} e^{-t} \, \mathrm{d}t, \qquad \text{(upper incomplete gamma function)}$$

$$\gamma(\alpha, x) \overset{\text{def.}}{=} \int_0^x t^{\alpha-1} e^{-t} \, \mathrm{d}t. \qquad \text{(lower incomplete gamma function)}$$

Clearly, their sum is the Gamma function: $\gamma(\alpha, x) + \Gamma(\alpha, x) = \Gamma(\alpha)$.

Using the lower incomplete gamma function, we can write the cdf of the Gamma distribution as:

$$\mathrm{F}\left[x \; ; \; \text{Gamma}(\alpha, \beta)\right] = \frac{\gamma(\alpha, \beta x)}{\Gamma(\alpha)}.$$

The support of Gamma can be restricted to $(0, 1)$, obtaining the **Truncated Gamma** distribution:

$$\text{Gamma}_{(0,1)}\left(x \; ; \; \alpha, \beta\right) \overset{\text{def.}}{=} \mathbb{1}_{(0,1)}(x) \times \frac{\text{Gamma}\left(x \; ; \; \alpha, \beta\right)}{\mathrm{F}\left[1 \; ; \; \text{Gamma}(\alpha, \beta)\right]} = \mathbb{1}_{(0,1)}(x) \times \frac{\beta^\alpha}{\gamma(\alpha, \beta)} x^{\alpha-1} e^{-\beta x}. \tag{2.2}$$

where $\alpha \in \mathbb{R}_{>0}$ and $\beta \in \mathbb{R}_{>0}$ are the *shape* and *rate* of the distribution.

How do we sample from the truncated Gamma distribution? We can use the inverse transform method, which consists of sampling from the uniform distribution and applying the inverse of the cdf of the truncated Gamma. Suppose that F is the cdf of the Gamma distribution and G is the cdf of the truncated Gamma distribution. We have that:

$$\forall x \in (0, 1), G(x) = \frac{F(x)}{F(1)} = \frac{\gamma(\alpha, \beta x)}{\gamma(\alpha, \beta)}.$$

Hence, the inverse of the cdf of the truncated Gamma is given by:

$$G^{-1}(y) = F^{-1}(F(1)y).$$

Algorithm 2 shows how to sample from the truncated Gamma distribution.

Algorithm 2: Sample from the Truncated Gamma Distribution

```
**Input:** Shape $\alpha$, rate $\beta$
**Output:** A sample $x \in (0, 1)$ from the Truncated Gamma distribution.
Sample $u \sim \text{Uniform}(0, 1)$.
**return** $F^{-1}(F(1)u)$
```

Let us compute the expectation of the truncated Gamma over $(0, 1)$:

$$\mathbb{E}\left[\text{Gamma}_{(0,1)}(\alpha, \beta)\right] = \int_0^1 x \cdot \text{Gamma}_{(0,1)}\left(x \; ; \; \alpha, \beta\right) \mathrm{d}x = \int_0^1 \frac{1}{\gamma(\alpha, \beta)} \cdot \beta^\alpha x^\alpha e^{-\beta x} \, \mathrm{d}x$$

$$\begin{aligned}&\overset{\beta x=t}{=} \frac{1}{\gamma(\alpha,\beta)}\int_0^\beta \frac{1}{\beta}t^\alpha e^{-t}\,\mathrm{d}t = \frac{1}{\beta}\cdot\frac{1}{\gamma(\alpha,\beta)}\cdot\int_0^\beta t^\alpha e^{-t}\,\mathrm{d}t\\ &= \frac{1}{\beta}\cdot\frac{\gamma(\alpha+1,\beta)}{\gamma(\alpha,\beta)} = \frac{\alpha}{\beta}\cdot\frac{\gamma(\alpha+1,\beta)\;/\;\Gamma(\alpha+1)}{\gamma(\alpha,\beta)\;/\;\Gamma(\alpha)}.\end{aligned}$$

Thus, we get the following formula:

$$\mathbb{E}\left[\mathrm{Gamma}_{(0,1)}(\alpha,\beta)\right] = \frac{1}{\beta}\cdot\frac{\gamma(\alpha+1,\beta)}{\gamma(\alpha,\beta)} = \frac{\alpha}{\beta}\cdot\frac{\overline{\gamma}(\alpha+1,\beta)}{\overline{\gamma}(\alpha,\beta)}. \tag{2.3}$$

where $\overline{\gamma}(a,z) \overset{\text{def.}}{=} \gamma(a,z)/\Gamma(a)$ is the regularized incomplete gamma function. We keep the second form of the expectation since some math libraries only provide the regularized form of the incomplete gamma function. When compared to the Monte Carlo estimate (not shown), the formula proves to be accurate.

How do we compute the expectation of the log-truncated Gamma distribution? Of course, we can use the previously defined sampling method to find a Monte Carlo estimate. However, it is also possible to find a formula for it. In order to do so, we need some results.

Adapting the findings from [Casella and Berger, 2002, pp. 111-112], in an exponential family distribution of the form $f_\theta(x) = \exp\{s(x)\theta - A(\theta) + h(x)\}$, the following relationship holds:

$$\mathbb{E}_\theta[s(X)] = A'(\theta).$$

Let us try to write the truncated Gamma distribution in such a form. We have:

$$\forall x\in(0,1), \mathrm{Gamma}_{(0,1)}\left(x\;;\;\alpha,\beta\right) = \frac{\beta^\alpha}{\gamma(\alpha,\beta)}x^{\alpha-1}e^{-\beta x} = \exp\left\{\alpha\log\beta - \log\gamma(\alpha,\beta) + (\alpha-1)\log x - \beta x\right\}.$$

Assuming that β is fixed, this is clearly an exponential family with $\theta=\alpha$, $s(x)=\log x$, $A(\theta)=\log\gamma(\alpha,\beta) - \alpha\log\beta$, and $h(x) = -\beta x - \log x$. Keeping in mind that $\mathbb{E}_\theta[s(X)] = A'(\theta)$, we get:

$$\mathbb{E}\left[\log\mathrm{Gamma}_{(0,1)}(\alpha,\beta)\right] = \frac{\partial}{\partial\alpha}\left\{\log\gamma(\alpha,\beta) - \alpha\log\beta\right\} = \frac{\frac{\partial}{\partial\alpha}\gamma(\alpha,\beta)}{\gamma(\alpha,\beta)} - \log\beta.$$

The derivative of the incomplete gamma function [Wolfram Research, Inc., 2001] is given by:

$$\begin{aligned}\frac{\partial}{\partial a}\gamma(a,z) &= \gamma(a,z)\log(z) - \Gamma(a)^2 z^a\cdot {}_2\widetilde{F}_2(a,a;a+1,a+1;-z)\\ &= \gamma(a,z)\log(z) - \frac{z^a}{a^2}\cdot {}_2F_2(a,a;a+1,a+1;-z).\end{aligned}$$

where ${}_2F_2$ is the generalized hypergeometric function and ${}_2\widetilde{F}_2$ is the regularized version.

Let us try to prove this formula. We start by introducing all the definitions that we need:

$$(a)_n = \frac{\Gamma(n+a)}{\Gamma(a)} = a(a+1)\cdots(a+n-1), \qquad \text{(the raising factorial - Pochhammer symbol)}$$

$${}_pF_q(a_1,\ldots,a_p;b_1,\ldots,b_q;z) = \sum_{n=0}^\infty \frac{(a_1)_n\cdots(a_p)_n}{(b_1)_n\cdots(b_q)_n}\cdot\frac{z^n}{n!}, \qquad \text{(generalized hypergeometric function)}$$

$$\widetilde{F}_q(a_1,\ldots,a_p;b_1,\ldots,b_q;z) = \frac{{}_pF_q(a_1,\ldots,a_p;b_1,\ldots,b_q;z)}{\Gamma(b_1)\cdots\Gamma(b_q)}. \qquad \text{(regularized gen. hypergeometric function)}$$

In particular:

$${}_2F_2(a,a;a+1,a+1;z) = \sum_{n=0}^\infty \frac{(a)_n^2}{(a+1)_n^2}\cdot\frac{(-z)^n}{n!} = \sum_{n=0}^\infty \frac{a^2}{n!(n+a)^2}\cdot(-1)^n z^n.$$

Using the power series expansion of e^{-t}, we obtain:

$$\begin{aligned}
\gamma(a,z) &= \int_0^z t^{a-1} \sum_{n=0}^{\infty} \frac{(-1)^n t^n}{n!}\,\mathrm{d}t = \sum_{n=0}^{\infty} \int_0^z \frac{(-1)^n t^{n+a-1}}{n!}\,\mathrm{d}t = \sum_{n=0}^{\infty} \frac{(-1)^n z^{n+a}}{n!(n+a)}, \\
\frac{\partial}{\partial a}\gamma(a,z) &= \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \cdot \frac{\partial}{\partial a}\left(\frac{z^{n+a}}{n+a}\right) = \sum_{n=0}^{\infty} \frac{(-1)^n}{n!}\left(\frac{z^{n+a}\log z}{n+a} - \frac{z^{n+a}}{(n+a)^2}\right) \\
&= \log z \sum_{n=0}^{\infty} \frac{(-1)^n z^{n+a}}{n!(n+a)} - \sum_{n=0}^{\infty} \frac{(-1)^n z^{n+a}}{n!(n+a)^2} = \gamma(a,z)\log z - z^a \sum_{n=0}^{\infty} \frac{(-1)^n z^n}{n!(n+a)^2} \\
&= \gamma(a,z)\log z - \frac{z^a}{a^2} \cdot {}_2F_2(a,a;a+1,a+1;-z).
\end{aligned}$$

Thus, we have proven the formula for the derivative of the incomplete gamma function. Now we can use it to compute the expectation of the log-truncated Gamma distribution. By plugging the derivative into the formula, we get:

$$\mathbb{E}\left[\log \mathrm{Gamma}_{(0,1)}(\alpha,\beta)\right] = (-1) \cdot \frac{\beta^\alpha \cdot {}_2F_2(\alpha,\alpha;\alpha+1,\alpha+1;-\beta)}{\alpha^2\gamma(\alpha,\beta)}. \tag{2.4}$$

We now have a formula for the expectation of the log-truncated Gamma distribution. When compared to the Monte Carlo estimate (not shown), the formula proves to be accurate.

2.6 Poisson Distribution

2.6.1 Poisson

The Poisson distribution is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time or space. The Poisson distribution is defined as

$$\mathrm{Poisson}(k;\lambda) = \mathbb{1}_{\mathbb{N}_{\geq 0}}(k) \times \frac{\lambda^k e^{-\lambda}}{k!},$$

where $k \in \mathbb{N}_{\geq 0}$ is a non-negative integer and $\lambda \in \mathbb{R}_{>0}$ is a positive real number. Its expectation is given by:

$$\mathbb{E}\left[\mathrm{Poisson}(\lambda)\right] = \lambda.$$

2.6.2 Positive Poisson (Zero-truncated Poisson)

The support of Poisson can be restricted to $\mathbb{N}_{>0}$, obtaining the **Positive Poisson** or **Zero-truncated Poisson** distribution, denoted by $\mathrm{Poisson}_+$:

$$\mathrm{Poisson}_+(k;\lambda) \overset{\text{def.}}{=} \mathbb{1}_{\mathbb{N}_{>0}}(k) \times \frac{\mathrm{Poisson}(k;\lambda)}{1-\mathrm{Poisson}(0;\lambda)} = \mathbb{1}_{\mathbb{N}_{>0}}(k) \times \frac{e^{-\lambda}}{1-e^{-\lambda}} \cdot \frac{\lambda^k}{k!},$$

where $k \in \mathbb{N}_{\geq 0}$ is a positive integer and $\lambda \in \mathbb{R}_{>0}$ is a positive real number. Its expectation is given by:

$$\mathbb{E}\left[\mathrm{Poisson}_+(\lambda)\right] = \frac{\lambda}{1-e^{-\lambda}}.$$

Chapter 3

Model

We are going to introduce our model in the context of recommendation systems, such as those used by online streaming services, e-commerce platforms, or social media sites. Of course, the model can be applied to other domains as well, such as cyber-link prediction. Such interpretations follow trivially from the model's structure.

In a recommendation system, we want to predict whether a user i will interact with a product j at time t. This interaction can be clicking on a product, watching a video, or any other user action that indicates interest. Let us see how we can model this problem:

Consider a set of times $\{1, \cdots, T\}$, where $t \in \{1, \cdots, T\}$ denotes the time index. Let V_1 be the set of N_1 users we have data for, with $i \in \{1, \cdots, N_1\}$ denoting the user index. Similarly, let V_2 be the set of N_2 products we have data for, with $j \in \{1, \cdots, N_2\}$ denoting the product index.

We use the matrix $\mathbf{A}$ to represent the interactions between users and products. Specifically, if $A_{tij} = 1$, then user i interacted with product j at time t; otherwise, if $A_{tij} = 0$, then user i did not interact with product j at time t.

Additionally, the matrix $\mathbf{N}$ represents the total number of interactions between users and products at each time. More precisely, N_{tij} denotes the number of interactions between user i and product j at time t.

We may have a lot of information about the users and products that can be used to make predictions. However, not all of this information is necessarily relevant, and high-dimensional data can be computationally expensive to process.

Therefore, it is often advantageous to derive a lower-dimensional representation of the data that captures the essential characteristics and relationships of the users and products. This representation is referred to as **latent position features**, denoted as $\mathbf{x}_i \in \mathbb{R}_+^d$ for user i and $\mathbf{y}_j \in \mathbb{R}_+^d$ for product j. These latent features are unobservable variables that influence the interactions between users and products.

Notice that the number of latent features for a user must be the same as the number of latent features for a product, denoted as d. We use $r \in \{1, \cdots, d\}$ to represent the latent feature index.

The only observations that we have are the occurent of interactions, that is, $\{\mathbf{A}_t\}$. We do not have access to the number of interactions between users and products.

Putting everything together, we are now ready to introduce our model. Here are its key components:

- **Time-dependent Correction Factors**:

$$\rho_{ti}^{(x)} \sim \text{Gamma}_{(0,1)}(\alpha^{(x)}, \beta^{(x)}) \quad \text{and} \quad \rho_{tj}^{(y)} \sim \text{Gamma}_{(0,1)}(\alpha^{(y)}, \beta^{(y)}).$$

  These factors modulate the overall intensity of interactions at different times. For example, a user i might have been inactive at certain times, so the time correction factor $\rho_{ti}^{(x)}$ would be very close to

0. Similarly, a product j might have been popular at certain times, so the time correction factor $\rho_{tj}^{(y)}$ would be very close to 1.
We use a truncated Gamma distribution to model these factors, primarily because this will give us a closed form solution for the posterior distribution, making the updates in the variational inference algorithm easier to compute. This choice is motivated by the fact that Gamma is a conjugate prior for a Poisson likelihood function. Note that we truncate the Gamma distribution in order to make sure that the correction factors are in the interval $(0, 1)$.

- **Time-invariant Latent Features**:

$$x_{ir} \mid \zeta_i^{(x)} \sim \text{Gamma}(a^{(x)}, \zeta_i^{(x)}) \text{ and } y_{jr} \mid \zeta_j^{(y)} \sim \text{Gamma}(a^{(y)}, \zeta_j^{(y)}).$$

These are the time-invariant latent features of the users and products. They capture the characteristics of users and products that are relevant for predicting interactions, but do not change over time. For example, a user i might have a high latent feature value if they are interested in a particular genre of movies, and a product j might have a high latent feature value if it belongs to that genre.
These features are modeled as Gamma distributions for the same reason as before: we will have a closed form solution for the posterior distribution.

- **Spread Hyperparameters**:

$$\zeta_i^{(x)} \sim \text{Gamma}(b^{(x)}, c^{(x)}) \text{ and } \zeta_j^{(y)} \sim \text{Gamma}(b^{(y)}, c^{(y)}).$$

These are hyperparameters that allow us to control the spread of the latent features, since they dictate the rate parameters of the Gamma distributions from which the latent features are drawn. A lower rate leads to a more spread-out distribution, while a higher rate leads to a more concentrated distribution. As before, Gamma is used as the prior distribution for these hyperparameters because it will give us a closed form solution for the posterior distribution.

- **Total Count**:

$$N_{tij} \mid \rho_{ti}^{(x)}, \rho_{tj}^{(y)}, \mathbf{x}_i, \mathbf{y}_j \sim \text{Poisson}(\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j).$$

This it the total number of interactions between user i and product j at time t. Since this is a non-negative integer, we model it as a Poisson distribution with a rate parameter that depends on the latent features and the scaling factors.

- **Contributions to the Total Count**:

$$N_{tij} = \sum_{r=1}^{d} Z_{tijr} \text{ and } Z_{tijr} \mid x_{ir}, y_{jr}, \rho_{ti}^{(x)}, \rho_{tj}^{(y)} \sim \text{Poisson}(\rho_{ti}^{(x)} \rho_{tj}^{(y)} x_{ir} y_{jr}).$$

Every latent feature contributes to the total count, and the total count is the sum of the contributions from all the latent features. We model the contribution of each latent feature as a Poisson distribution with a rate parameter that depends on the latent features and the scaling factors.

- **Interaction Matrix**:

$$A_{tij} = \mathbb{1}_{\mathbb{N}_+}(N_{tij}).$$

This is the matrix of interactions between users and products. It is a binary matrix, where $A_{tij} = 1$ if the user interacted with the product at time t, and $A_{tij} = 0$ otherwise. Since N is modeled as a Poisson distribution, the interactions are modeled as a Bernoulli distribution:

$$A_{tij} \mid \rho_{ti}^{(x)}, \rho_{tj}^{(y)}, \mathbf{x}_i, \mathbf{y}_j \sim \text{Bernoulli}\left[1 - \exp(-\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j)\right].$$

By putting everything together, we get the following model:

$$\begin{aligned} A_{tij} &= \mathbb{1}_{\mathbb{N}_+}(N_{tij}), \\ N_{tij} \mid \mathbf{x}_i, \mathbf{y}_j, \rho_{ti}^{(x)}, \rho_{tj}^{(y)} &\sim \mathrm{Poisson}(\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j), \\ x_{ir} \mid \zeta_i^{(x)} &\sim \mathrm{Gamma}(a^{(x)}, \zeta_i^{(x)}), \\ y_{jr} \mid \zeta_j^{(y)} &\sim \mathrm{Gamma}(a^{(y)}, \zeta_j^{(y)}), \\ \zeta_i^{(x)} &\sim \mathrm{Gamma}(b^{(x)}, c^{(x)}), \\ \zeta_j^{(y)} &\sim \mathrm{Gamma}(b^{(y)}, c^{(y)}), \\ \rho_{ti}^{(x)} &\sim \mathrm{Gamma}_{(0,1)}(\alpha^{(x)}, \beta^{(x)}), \\ \rho_{tj}^{(y)} &\sim \mathrm{Gamma}_{(0,1)}(\alpha^{(y)}, \beta^{(y)}), \end{aligned}$$

where $a^{(x)}$, $a^{(y)}$, $b^{(x)}$, $b^{(y)}$, $c^{(x)}$, $c^{(y)}$, $\alpha^{(x)}$, $\alpha^{(y)}$, $\beta^{(x)}$, $\beta^{(y)} \in \mathbb{R}_+$ are positive deterministic constants. A graphical representation of the model is shown in Figure 3.1.

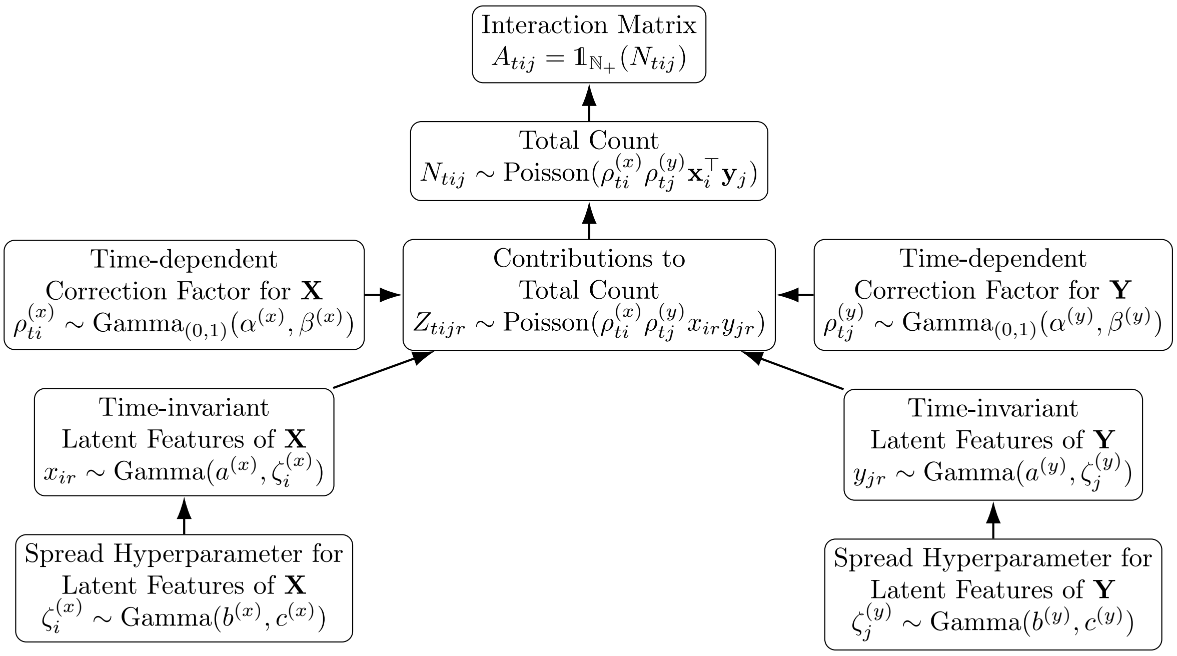

Interaction Matrix
$A_{tij} = \mathbb{1}_{\mathbb{N}_+}(N_{tij})$
Total Count
$N_{tij} \sim \mathrm{Poisson}(\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j)$
Time-dependent Correction Factor for $\mathbf{X}$
$\rho_{ti}^{(x)} \sim \mathrm{Gamma}_{(0,1)}(\alpha^{(x)}, \beta^{(x)})$
Contributions to Total Count
$Z_{tijr} \sim \mathrm{Poisson}(\rho_{ti}^{(x)} \rho_{tj}^{(y)} x_{ir} y_{jr})$
Time-dependent Correction Factor for $\mathbf{Y}$
$\rho_{tj}^{(y)} \sim \mathrm{Gamma}_{(0,1)}(\alpha^{(y)}, \beta^{(y)})$
Time-invariant Latent Features of $\mathbf{X}$
$x_{ir} \sim \mathrm{Gamma}(a^{(x)}, \zeta_i^{(x)})$
Time-invariant Latent Features of $\mathbf{Y}$
$y_{jr} \sim \mathrm{Gamma}(a^{(y)}, \zeta_j^{(y)})$
Spread Hyperparameter for Latent Features of $\mathbf{X}$
$\zeta_i^{(x)} \sim \mathrm{Gamma}(b^{(x)}, c^{(x)})$
Spread Hyperparameter for Latent Features of $\mathbf{Y}$
$\zeta_j^{(y)} \sim \mathrm{Gamma}(b^{(y)}, c^{(y)})$

Figure 3.1: Diagram of the Proposed Model

The only observations are $\{\mathbf{A}_t\}$. All the other random variables are parameters that need to be inferred. Note that $\mathbf{Z}$ and $\mathbf{N}$ are auxiliary variables (used for inference); they are not strictly part of the model.

It is straightforward to notice that $\mathbf{A}$ follows a Bernoulli distribution with $p(A_{tij} = 0) = p(N_{tij} = 0)$:

$$A_{tij} \mid \rho_{ti}^{(x)}, \rho_{tj}^{(y)}, \mathbf{x}_i, \mathbf{y}_j \sim \mathrm{Bernoulli}\left[1 - \exp(-\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j)\right].$$

As we have mentioned before, in order to simplify our calculations, we are going to keep track of the contribution of each component to the total latent count by introducing a new set of random variables $\mathbf{Z} = \{Z_{tijr}\}$, where:

$$N_{tij} = \sum_{r=1}^{d} Z_{tijr} \quad \text{and} \quad Z_{tijr} \mid x_{ir}, y_{jr}, \rho_{ti}^{(x)}, \rho_{tj}^{(y)} \sim \mathrm{Poisson}(\rho_{ti}^{(x)} \rho_{tj}^{(y)} x_{ir} y_{jr}).$$

Furthermore, let us consider the set of parameters for this model:

$$\mathcal{P} \overset{\text{def.}}{=} \Big\{ N_{tij}, Z_{tijr}, x_{ir}, y_{jr}, \zeta_i^{(x)}, \zeta_j^{(y)}, \rho_{ti}^{(x)}, \rho_{tj}^{(y)} \\ \Big| \; t \in \{1, \cdots, T\}, i \in \{1, \cdots, N_1\}, j \in \{1, \cdots, N_2\}, r \in \{1, \cdots, d\} \Big\}.$$

To fit this model, we are going to use variational inference with CAVI (Algorithm 1), which is a more scalable alternative to MCMC methods. We will discuss the details of the variational inference algorithm in the next sections.

3.1 Likelihood

Given the observations $\{\mathbf{A}_t\}$, we aim to compute the likelihood of the parameters. However, $\mathbf{Z}$ and $\mathbf{N}$ are auxiliary parameters introduced to simplify computations in the variational inference algorithm. Therefore, when calculating the likelihood, we can disregard them. Let $\Phi \overset{\text{def.}}{=} \mathcal{P} \setminus \{\mathbf{Z}, \mathbf{N}\}$ represent the set of all parameters except $\mathbf{Z}$ and $\mathbf{N}$.

Then we can write the likelihood and log likelihood as:

$$\begin{aligned}
L(\Phi \mid \mathbf{A}) = p(\mathbf{A} \mid \Phi) &= \prod_{t=1}^{T} \prod_{i=1}^{N_1} \prod_{j=1}^{N_2} p(A_{tij} \mid \Phi) \\
&= \prod_{t=1}^{T} \prod_{i=1}^{N_1} \prod_{j=1}^{N_2} \left\{ 1 - \exp\left(-\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j \right) \right\}^{A_{tij}} \exp\left(-\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j \right)^{1 - A_{tij}} \\
&= \prod_{t=1}^{T} \prod_{i=1}^{N_1} \prod_{j=1}^{N_2} \left\{ \frac{1 - \exp\left(-\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j \right)}{\exp\left(-\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j \right)} \right\}^{A_{tij}} \exp\left(-\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j \right) \\
&= \prod_{t=1}^{T} \prod_{i=1}^{N_1} \prod_{j=1}^{N_2} \left\{ \exp\left(\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j \right) - 1 \right\}^{A_{tij}} \exp\left(-\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j \right) \\
&= \prod_{t,i,j : A_{tij} = 1} \left\{ \exp\left(\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j \right) - 1 \right\} \times \prod_{t=1}^{T} \prod_{i=1}^{N_1} \prod_{j=1}^{N_2} \exp\left(-\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j \right), \\
\log L(\Phi \mid \mathbf{A}) &= \sum_{t,i,j : A_{tij} = 1} \log \left\{ \exp\left(\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j \right) - 1 \right\} - \sum_{t=1}^{T} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j.
\end{aligned}$$

Let us examine the time complexity of the log likelihood. The number of latent features can be considered a constant, as it is a hyperparameter, and ideally, we aim to use as few features as possible. Therefore, we can exclude it from our analysis.

The first sum has time complexity $O[\text{nnz}(\mathbf{A})]$, where $\text{nnz}(\mathbf{A})$ is the number of non-zero elements in $\mathbf{A}$.

The second sum seems to have complexity $O[TN_1N_2]$, but we can reduce it to $O[T(N_1 + N_2)]$ by using the following trick:

$$\sum_{t=1}^{T} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j = \sum_{t=1}^{T} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sum_{r=1}^{d} \rho_{ti}^{(x)} \rho_{tj}^{(y)} x_{ir} y_{jr} = \sum_{t=1}^{T} \sum_{r=1}^{d} \left(\sum_{i=1}^{N_1} \rho_{ti}^{(x)} x_{ir} \right) \left(\sum_{j=1}^{N_2} \rho_{tj}^{(y)} y_{jr} \right).$$

Hence, the time complexity of the likelihood is $O[\text{nnz}(\mathbf{A}) + T(N_1 + N_2)]$.

In most statistical models, evaluating the likelihood has a complexity of $O(N_1N_2)$, which makes our model especially attractive for large, sparse networks where $\text{nnz}(\mathbf{A}) \ll N_1N_2$. Such networks are commonly encountered in fields like cybersecurity, social networks, and biological networks.

3.2 Posterior Distribution

In order to compute the full complete conditionals of the parameters, we start by computing the conditional of all parameters given the observations, that is:

$$p(\mathcal{P} \mid \mathbf{A}) = p\left(\boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \mathbf{x}, \mathbf{y}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}, \mathbf{Z}, \mathbf{N} \,\middle|\, \mathbf{A}\right).$$

If at least one of the conditions $A_{tij} = \mathbb{1}_{\mathbb{N}_+}(N_{tij})$ and $N_{tij} = \sum_{r=1}^{d} Z_{tijr}$ does not hold, then the joint distribution $p(\mathcal{P}, \mathbf{A})$ is 0. Therefore, we need to introduce this condition in our calculations.

$$\begin{aligned}
p(\mathcal{P} \mid \mathbf{A}) &\propto p\left(\boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}, \mathbf{x}, \mathbf{y}, \mathbf{Z}, \mathbf{N}, \mathbf{A}\right) \\
&= \prod_{t=1}^{T}\prod_{i=1}^{N_1}\prod_{j=1}^{N_2} \mathbb{1}\left[A_{tij} = \mathbb{1}_{\mathbb{N}_+}(N_{tij})\right] \times \prod_{t=1}^{T}\prod_{i=1}^{N_1}\prod_{j=1}^{N_2} \mathbb{1}\left[N_{tij} = \sum_{r=1}^{d} Z_{tijr}\right] \\
&\quad \times p\left(\boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}, \mathbf{x}, \mathbf{y}, \mathbf{Z}\right).
\end{aligned}$$

Now let us compute the joint $p\left(\boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}, \mathbf{x}, \mathbf{y}, \mathbf{Z}\right)$:

$$\begin{aligned}
&p\left(\boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}, \mathbf{x}, \mathbf{y}, \mathbf{Z}\right) \\
&= p\left(\mathbf{Z} \,\middle|\, \boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}, \mathbf{x}, \mathbf{y}\right) \\
&\quad \times p\left(\mathbf{x} \,\middle|\, \boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}, \mathbf{y}\right) \times p\left(\mathbf{y} \,\middle|\, \boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}\right) \times p\left(\boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}\right) \\
&= p\left(\mathbf{Z} \,\middle|\, \boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \mathbf{x}, \mathbf{y}\right) \times p\left(\mathbf{x} \,\middle|\, \boldsymbol{\zeta}^{(x)}\right) \times p\left(\mathbf{y} \,\middle|\, \boldsymbol{\zeta}^{(y)}\right) \times p\left(\boldsymbol{\zeta}^{(x)}\right) \times p\left(\boldsymbol{\zeta}^{(y)}\right) \times p\left(\boldsymbol{\rho}^{(x)}\right) \times p\left(\boldsymbol{\rho}^{(y)}\right).
\end{aligned}$$

Hence, the conditional of all parameters given the observations is given by:

$$\begin{aligned}
p(\mathcal{P} \mid \mathbf{A}) &\propto p\left(\boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}, \mathbf{x}, \mathbf{y}, \mathbf{Z}, \mathbf{N}, \mathbf{A}\right) \\
&\propto \prod_{t=1}^{T}\prod_{i=1}^{N_1}\prod_{j=1}^{N_2} \mathbb{1}\left[A_{tij} = \mathbb{1}_{\mathbb{N}_+}(N_{tij})\right] \times \prod_{t=1}^{T}\prod_{i=1}^{N_1}\prod_{j=1}^{N_2} \mathbb{1}\left[N_{tij} = \sum_{r=1}^{d} Z_{tijr}\right] \\
&\quad \times \prod_{t=1}^{T}\prod_{i=1}^{N_1}\prod_{j=1}^{N_2}\prod_{r=1}^{d} (Z_{tijr}!)^{-1} \cdot \left(\rho_{ti}^{(x)}\rho_{tj}^{(y)} x_{ir} y_{jr}\right)^{Z_{tijr}} \cdot \exp\left\{-\rho_{ti}^{(x)}\rho_{tj}^{(y)} x_{ir} y_{jr}\right\} \\
&\quad \times \prod_{i=1}^{N_1}\prod_{r=1}^{d} \left\{\zeta_i^{(x)}\right\}^{a^{(x)}} \cdot \{x_{ir}\}^{a^{(x)}-1} \cdot \exp\left\{-x_{ir}\zeta_i^{(x)}\right\} \\
&\quad \times \prod_{j=1}^{N_2}\prod_{r=1}^{d} \left\{\zeta_j^{(y)}\right\}^{a^{(y)}} \cdot \{y_{jr}\}^{a^{(y)}-1} \cdot \exp\left\{-y_{jr}\zeta_j^{(y)}\right\} \\
&\quad \times \prod_{i=1}^{N_1} \left\{\zeta_i^{(x)}\right\}^{b^{(x)}-1} \cdot \exp\left\{-c^{(x)}\zeta_i^{(x)}\right\} \\
&\quad \times \prod_{j=1}^{N_2} \left\{\zeta_j^{(y)}\right\}^{b^{(y)}-1} \cdot \exp\left\{-c^{(y)}\zeta_j^{(y)}\right\} \\
&\quad \times \prod_{t=1}^{T}\prod_{i=1}^{N_1} \left\{\rho_{ti}^{(x)}\right\}^{\alpha^{(x)}-1} \cdot \exp\left\{-\beta^{(x)}\rho_{ti}^{(x)}\right\} \\
&\quad \times \prod_{t=1}^{T}\prod_{j=1}^{N_2} \left\{\rho_{tj}^{(y)}\right\}^{\alpha^{(y)}-1} \cdot \exp\left\{-\beta^{(y)}\rho_{tj}^{(y)}\right\}.
\end{aligned}$$

Since we have the full conditional distribution of the parameters given the observations, we are now ready to compute the complete conditional distributions.

3.3 Complete conditional distributions

Using the previously computed form of the full conditional distribution, we obtain the following conditionals:

$$\begin{aligned}
&p\left(\mathbf{Z}_{tij}, N_{tij} \mid \mathcal{P} \setminus \{\mathbf{Z}_{tij}, N_{tij}\}, \mathbf{A}\right) \\
&\propto \mathbb{1}\left[A_{tij} = \mathbb{1}_{\mathbb{N}_+}(N_{tij})\right] \times \mathbb{1}\left[N_{tij} = \sum_{r=1}^{d} Z_{tijr}\right] \times \prod_{r=1}^{d} (Z_{tijr}!)^{-1} \cdot \left(\rho_{ti}^{(x)} \rho_{tj}^{(y)} x_{ir} y_{jr}\right)^{Z_{tijr}} \\
&= \mathbb{1}\left[A_{tij} = \mathbb{1}_{\mathbb{N}_+}(N_{tij})\right] \times \mathbb{1}\left[N_{tij} = \sum_{r=1}^{d} Z_{tijr}\right] \\
&\qquad \times (N_{tij}!)^{-1} \left(\rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j\right)^{N_{tij}} \times \frac{N_{tij}!}{\prod_{r=1}^{d} Z_{tijr}!} \prod_{r=1}^{d} \left(\frac{x_{ir} y_{jr}}{\mathbf{x}_i^\top \mathbf{y}_j}\right)^{Z_{tijr}} \\
&\propto \mathbb{1}\left[A_{tij} = \mathbb{1}_{\mathbb{N}_+}(N_{tij})\right] \times \text{Poisson}_+\left(N_{tij} \ ; \ \rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j\right) \times \text{Multinomial}\left(\mathbf{Z}_{tij} \ ; \ N_{tij}, \left\{\frac{x_{ir} y_{jr}}{\mathbf{x}_i^\top \mathbf{y}_j}\right\}_{r=1}^{d}\right).
\end{aligned}$$

Notice that here we use a Poisson$_+$ distribution instead of Poisson since $\mathbf{A}$ sets the support of the conditional $Z_{tijr}, N_{tij} \mid \mathcal{P} \setminus \{Z_{tijr}, N_{tij}\}, \mathbf{A}$. If $A_{tij} = 0$, then N_{tij} can only be 0. If $A_{tij} = 1$, then N_{tij} must be a positive integer, so it follows a Poisson$_+$ distribution.

$$\begin{aligned}
p\left(x_{ir} \mid \mathcal{P} \setminus \{x_{ir}\}, \mathbf{A}\right) &\propto \{x_{ir}\}^{a^{(x)} - 1 + \sum_{t=1}^{T} \sum_{j=1}^{N_2} Z_{tijr}} \cdot \exp\left\{-x_{ir}\left[\zeta_i^{(x)} + \sum_{t=1}^{T} \sum_{j=1}^{N_2} \rho_{ti}^{(x)} \rho_{tj}^{(y)} y_{jr}\right]\right\}, \\
p\left(y_{jr} \mid \mathcal{P} \setminus \{y_{jr}\}, \mathbf{A}\right) &\propto \{y_{jr}\}^{a^{(y)} - 1 + \sum_{t=1}^{T} \sum_{i=1}^{N_1} Z_{tijr}} \cdot \exp\left\{-y_{jr}\left[\zeta_j^{(y)} + \sum_{t=1}^{T} \sum_{i=1}^{N_1} \rho_{ti}^{(x)} \rho_{tj}^{(y)} x_{ir}\right]\right\}, \\
p\left(\zeta_i^{(x)} \mid \mathcal{P} \setminus \{\zeta_i^{(x)}\}, \mathbf{A}\right) &\propto \left\{\zeta_i^{(x)}\right\}^{d \cdot a^{(x)} + b^{(x)} - 1} \cdot \exp\left\{-\zeta_i^{(x)}\left[c^{(x)} + \sum_{r=1}^{d} x_{ir}\right]\right\}, \\
p\left(\zeta_j^{(y)} \mid \mathcal{P} \setminus \{\zeta_j^{(y)}\}, \mathbf{A}\right) &\propto \left\{\zeta_j^{(y)}\right\}^{d \cdot a^{(y)} + b^{(y)} - 1} \cdot \exp\left\{-\zeta_j^{(y)}\left[c^{(y)} + \sum_{r=1}^{d} y_{jr}\right]\right\}, \\
p\left(\rho_{ti}^{(x)} \mid \mathcal{P} \setminus \{\rho_{ti}^{(x)}\}, \mathbf{A}\right) &\propto \left\{\rho_{ti}^{(x)}\right\}^{\alpha^{(x)} - 1 + \sum_{j=1}^{N_2} N_{tij}} \cdot \exp\left\{-\rho_{ti}^{(x)}\left[\beta^{(x)} + \sum_{j=1}^{N_2} \sum_{r=1}^{d} \rho_{tj}^{(y)} x_{ir} y_{jr}\right]\right\}, \\
p\left(\rho_{tj}^{(y)} \mid \mathcal{P} \setminus \{\rho_{tj}^{(y)}\}, \mathbf{A}\right) &\propto \left\{\rho_{tj}^{(y)}\right\}^{\alpha^{(y)} - 1 + \sum_{i=1}^{N_1} N_{tij}} \cdot \exp\left\{-\rho_{tj}^{(y)}\left[\beta^{(y)} + \sum_{i=1}^{N_1} \sum_{r=1}^{d} \rho_{ti}^{(x)} x_{ir} y_{jr}\right]\right\}.
\end{aligned}$$

Therefore:

$$\begin{aligned}
&\mathbf{Z}_{tij}, N_{tij} \mid \mathcal{P} \setminus \{\mathbf{Z}_{tij}, N_{tij}\}, \mathbf{A} \\
&\quad = A_{tij} \times \text{Poisson}_+\left(N_{tij} \ ; \ \rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j\right) \times \text{Multinomial}\left(\mathbf{Z}_{tij} \ ; \ N_{tij}, \left\{\frac{x_{ir} y_{jr}}{\mathbf{x}_i^\top \mathbf{y}_j}\right\}_r\right) \\
&\quad \sim \begin{cases} \text{Poisson}_+\left(N_{tij} \ ; \ \rho_{ti}^{(x)} \rho_{tj}^{(y)} \mathbf{x}_i^\top \mathbf{y}_j\right) \times \text{Multinomial}\left(\mathbf{Z}_{tij} \ ; \ N_{tij}, \left\{\frac{x_{ir} y_{jr}}{\mathbf{x}_i^\top \mathbf{y}_j}\right\}_r\right) & \text{if } A_{tij} = 1, \\ \delta_0\left(\mathbf{Z}_{tij}, N_{tij}\right) & \text{if } A_{tij} = 0. \end{cases},
\end{aligned}$$

$$
\begin{aligned}
x_{ir} \mid \mathcal{P} \setminus \{x_{ir}\}, \mathbf{A} &\sim \text{Gamma}\left(a^{(x)} + \sum_{t=1}^{T}\sum_{j=1}^{N_2} Z_{tijr}\ ,\ \zeta_i^{(x)} + \sum_{t=1}^{T}\sum_{j=1}^{N_2} \rho_{ti}^{(x)}\rho_{tj}^{(y)} y_{jr}\right),\\
y_{jr} \mid \mathcal{P} \setminus \{y_{jr}\}, \mathbf{A} &\sim \text{Gamma}\left(a^{(y)} + \sum_{t=1}^{T}\sum_{i=1}^{N_1} Z_{tijr}\ ,\ \zeta_j^{(y)} + \sum_{t=1}^{T}\sum_{i=1}^{N_1} \rho_{ti}^{(x)}\rho_{tj}^{(y)} x_{ir}\right),\\
\zeta_i^{(x)} \mid \mathcal{P} \setminus \{\zeta_i^{(x)}\}, \mathbf{A} &\sim \text{Gamma}\left(d\cdot a^{(x)} + b^{(x)}\ ,\ c^{(x)} + \sum_{r=1}^{d} x_{ir}\right),\\
\zeta_j^{(y)} \mid \mathcal{P} \setminus \{\zeta_j^{(y)}\}, \mathbf{A} &\sim \text{Gamma}\left(d\cdot a^{(y)} + b^{(y)}\ ,\ c^{(y)} + \sum_{r=1}^{d} y_{jr}\right),\\
\rho_{ti}^{(x)} \mid \mathcal{P} \setminus \{\rho_{ti}^{(x)}\}, \mathbf{A} &\sim \text{Gamma}_{(0,1)}\left(\alpha^{(x)} + \sum_{j=1}^{N_2} N_{tij}\ ,\ \beta^{(x)} + \sum_{j=1}^{N_2}\sum_{r=1}^{d} \rho_{tj}^{(y)} x_{ir} y_{jr}\right),\\
\rho_{tj}^{(y)} \mid \mathcal{P} \setminus \{\rho_{tj}^{(y)}\}, \mathbf{A} &\sim \text{Gamma}_{(0,1)}\left(\alpha^{(y)} + \sum_{i=1}^{N_1} N_{tij}\ ,\ \beta^{(y)} + \sum_{i=1}^{N_1}\sum_{r=1}^{d} \rho_{ti}^{(x)} x_{ir} y_{jr}\right).
\end{aligned}
$$

We are now ready to propose a variational family.

3.4 Variational Family

Besides $\{\mathbf{A}_t\}$, all the other random variables are parameters that need to be estimated. Suggested by the form of our complete conditional distributions, we propose the following variational family:

$$
\begin{aligned}
q(\mathbf{Z}_{tij}, N_{tij}) &= A_{tij} \times \text{Poisson}_+\left(N_{tij}\ ;\ \phi_{tij}\right) \times \text{Multinomial}\left(\mathbf{Z}_{tij}\ ;\ N_{tij}, \boldsymbol{\chi}_{tij}\right)\\
&\sim \begin{cases} \text{Poisson}_+\left(N_{tij}\ ;\ \phi_{tij}\right) \times \text{Multinomial}\left(\mathbf{Z}_{tij}\ ;\ N_{tij}, \boldsymbol{\chi}_{tij}\right) & \text{if } A_{tij} = 1,\\ \delta_0\left(\mathbf{Z}_{tij}, N_{tij}\right) & \text{if } A_{tij} = 0. \end{cases},\\
q(x_{ir}) &\sim \text{Gamma}\left(\lambda_{ir}^{(x)}, \mu_{ir}^{(x)}\right),\\
q(y_{jr}) &\sim \text{Gamma}\left(\lambda_{jr}^{(y)}, \mu_{jr}^{(y)}\right),\\
q(\zeta_i^{(x)}) &\sim \text{Gamma}\left(\delta_i^{(x)}, \eta_i^{(x)}\right),\\
q(\zeta_j^{(y)}) &\sim \text{Gamma}\left(\delta_j^{(y)}, \eta_j^{(y)}\right),\\
q(\rho_{ti}^{(x)}) &\sim \text{Gamma}_{(0,1)}\left(m_{ti}^{(x)}, n_{ti}^{(x)}\right),\\
q(\rho_{tj}^{(y)}) &\sim \text{Gamma}_{(0,1)}\left(m_{tj}^{(y)}, n_{tj}^{(y)}\right).
\end{aligned}
$$

Note that such a choice for the distribution of $\mathbf{Z}_{tij}$ and N_{tij} makes sense: if we did not have any interaction between the nodes i and j at time t, then the values of $\mathbf{Z}_{tij}$ and N_{tij} would be zero. This is why we have the δ_0 distribution in the case when $A_{tij} = 0$.

Before proceeding with deriving the updates, we will first establish some properties of these distributions.

By integrating $\mathbf{Z}_{tij}$ from the joint distribution of $(\mathbf{Z}_{tij}, N_{tij})$, we get:

$$q(N_{tij}) = A_{tij} \times \text{Poisson}_+\left(N_{tij}\ ;\ \phi_{tij}\right),$$

which then gives:

$$q(\mathbf{Z}_{tij} \mid N_{tij}) = \text{Multinomial}\left(\mathbf{Z}_{tij}\ ;\ N_{tij}, \boldsymbol{\chi}_{tij}\right).$$

Since we have the closed form for these distributions, we are able to compute the expectations of N_{tij} and $\mathbf{Z}_{tij}$ as follows:

$$\mathbb{E}_q\left[N_{tij}\right] = A_{tij} \cdot \frac{\phi_{tij}}{1-\exp\{-\phi_{tij}\}},$$
$$\mathbb{E}_q\left[\mathbf{Z}_{tij}\right] = \mathbb{E}_{q(N_{tij})}\left[\mathbb{E}_{q(\mathbf{Z}_{tij}\mid N_{tij})}\left[\mathbf{Z}_{tij}\mid N_{tij}\right]\right] = \mathbb{E}_{q(N_{tij})}\left[N_{tij}\cdot\boldsymbol{\chi}_{tij}\right] = A_{tij}\cdot\frac{\phi_{tij}}{1-\exp\{-\phi_{tij}\}}\cdot\boldsymbol{\chi}_{tij}.$$

Moreover, let us define the following quantities for $q(P) \sim \text{Gamma}(\alpha,\beta)$ and $q(R) \sim \text{Gamma}_{(0,1)}(\alpha,\beta)$ from our variational inference model. Here, P can be x_{ir}, y_{jr}, $\zeta_i^{(x)}$, or $\zeta_j^{(y)}$, while R can be $\rho_{ti}^{(x)}$ or $\rho_{tj}^{(y)}$.

$$\text{ELG}\left[P\right] \overset{\text{def.}}{=} \mathbb{E}_q\left[\log P\right] = \mathbb{E}\left[\log\text{Gamma}(\alpha,\beta)\right] = \psi(\alpha)-\log(\beta),$$
$$\text{ETG}\left[R\right] \overset{\text{def.}}{=} \mathbb{E}_q\left[R\right] = \mathbb{E}\left[\text{Gamma}_{(0,1)}(\alpha,\beta)\right] = \frac{1}{\beta}\cdot\frac{\gamma(\alpha+1,\beta)}{\gamma(\alpha,\beta)},$$
$$\text{ELTG}\left[R\right] \overset{\text{def.}}{=} \mathbb{E}_q\left[\log R\right] = \mathbb{E}\left[\log\text{Gamma}_{(0,1)}(\alpha,\beta)\right] = (-1)\cdot\frac{\beta^\alpha\cdot{}_2F_2(\alpha,\alpha;\alpha+1,\alpha+1;-\beta)}{\alpha^2\gamma(\alpha,\beta)}.$$

These quantities are computed using (2.1), (2.3), and (2.4).

3.5 Updates

We are now ready to find the updates for each of the parameters in our variational family. Recall that we must take the expectations of the logarithms of the complete conditional distributions for each parameter, and then exponentiate these to derive the form of the variational distribution.

3.5.1 Counts

Keeping in mind that:

$$p\left(\mathbf{Z}_{tij}, N_{tij}\mid\mathcal{P}\setminus\{\mathbf{Z}_{tij},N_{tij}\},\mathbf{A}\right)$$
$$\propto \mathbb{1}\left[A_{tij}=\mathbb{1}_{\mathbb{N}_+}(N_{tij})\right]\times\mathbb{1}\left[N_{tij}=\sum_{r=1}^d Z_{tijr}\right]\times\prod_{r=1}^d\left(Z_{tijr}!\right)^{-1}\cdot\left(\rho_{ti}^{(x)}\rho_{tj}^{(y)}x_{ir}y_{jr}\right)^{Z_{tijr}},$$

we have:

$$\begin{aligned}\log q^*(\mathbf{Z}_{tij},N_{tij}) &\propto \mathbb{E}_{q(\mathcal{P}\setminus\{\mathbf{Z}_{tij},N_{tij}\})}\left[\log p\left(\mathbf{Z}_{tij},N_{tij}\mid\mathcal{P}\setminus\{\mathbf{Z}_{tij},N_{tij}\},\mathbf{A}\right)\right]\\
&\propto \mathbb{1}\left[A_{tij}=\mathbb{1}_{\mathbb{N}_+}(N_{tij})\right]\times\mathbb{1}\left[N_{tij}=\sum_{r=1}^d Z_{tijr}\right]\\
&\quad\times\mathbb{E}_{q(\mathcal{P}\setminus\{\mathbf{Z}_{tij},N_{tij}\})}\left[N_{tij}\log(\rho_{ti}^{(x)}\rho_{tj}^{(y)})+\sum_{r=1}^d\log(Z_{tijr}!^{-1})+Z_{tijr}\log(x_{ir}y_{jr})\right]\\
&\propto \mathbb{1}\left[A_{tij}=\mathbb{1}_{\mathbb{N}_+}(N_{tij})\right]\times\mathbb{1}\left[N_{tij}=\sum_{r=1}^d Z_{tijr}\right]\\
&\quad\times\left\{N_{tij}\cdot\left\{\text{ELTG}\left[\rho_{ti}^{(x)}\right]+\text{ELTG}\left[\rho_{tj}^{(y)}\right]\right\}+\sum_{r=1}^d\log(Z_{tijr}!^{-1})+Z_{tijr}\cdot\left\{\text{ELG}\left[x_{ir}\right]+\text{ELG}\left[y_{jr}\right]\right\}\right\},\end{aligned}$$

since the first two indicator functions establish the support of p, which consequently sets the supports of $\log p$ and $\log q^*$. By the same logic, they also determine the support of q^*, resulting in the following expression:

$$
\begin{aligned}
&q^*(\mathbf{Z}_{tij}, N_{tij}) \\
&\quad\propto \mathbb{1}\left[A_{tij} = \mathbb{1}_{\mathbb{N}_+}(N_{tij})\right] \times \mathbb{1}\left[N_{tij} = \sum_{r=1}^{d} Z_{tijr}\right] \\
&\qquad \times \exp\left\{\mathrm{ELTG}\left[\rho_{ti}^{(x)}\right] + \mathrm{ELTG}\left[\rho_{tj}^{(y)}\right]\right\}^{N_{tij}} \times \prod_{r=1}^{d} Z_{tijr}!^{-1} \cdot \exp\left\{\mathrm{ELG}\left[x_{ir}\right] + \mathrm{ELG}\left[y_{jr}\right]\right\}^{Z_{tijr}} \\
&\quad\propto \mathbb{1}\left[A_{tij} = \mathbb{1}_{\mathbb{N}_+}(N_{tij})\right] \times \mathbb{1}\left[N_{tij} = \sum_{r=1}^{d} Z_{tijr}\right] \\
&\qquad \times \left\{\exp\left\{\mathrm{ELTG}\left[\rho_{ti}^{(x)}\right] + \mathrm{ELTG}\left[\rho_{tj}^{(y)}\right]\right\} \cdot \sum_{r=1}^{d} \exp\left\{\mathrm{ELG}\left[x_{ir}\right] + \mathrm{ELG}\left[y_{jr}\right]\right\}\right\}^{N_{tij}} \\
&\qquad \times \prod_{r=1}^{d} Z_{tijr}!^{-1} \cdot \left\{\frac{\exp\left\{\mathrm{ELG}[x_{ir}] + \mathrm{ELG}[y_{jr}]\right\}}{\sum_{r=1}^{d} \exp\left\{\mathrm{ELG}[x_{ir}] + \mathrm{ELG}[y_{jr}]\right\}}\right\}^{Z_{tijr}}.
\end{aligned}
$$

This is the form of a Poisson$_+$-Multinomial distribution:

$$
\begin{aligned}
q^*(\mathbf{Z}_{tij}, N_{tij}) &= A_{tij} \times \mathrm{Poisson}_+\left(N_{tij}\ ;\ \phi_{tij}^{\mathrm{new}}\right) \times \mathrm{Multinomial}\left(\mathbf{Z}_{tij}\ ;\ N_{tij}, \boldsymbol{\chi}_{tij}^{\mathrm{new}}\right) \\
&\sim \begin{cases} \mathrm{Poisson}_+\left(N_{tij}\ ;\ \phi_{tij}^{\mathrm{new}}\right) \times \mathrm{Multinomial}\left(\mathbf{Z}_{tij}\ ;\ N_{tij}, \boldsymbol{\chi}_{tij}^{\mathrm{new}}\right) & \text{if } A_{tij} = 1, \\ \delta_0\left(\mathbf{Z}_{tij}, N_{tij}\right) & \text{if } A_{tij} = 0. \end{cases}, \\
&\text{where } \phi_{tij}^{\mathrm{new}} \overset{\text{def.}}{=} \exp\left\{\mathrm{ELTG}\left[\rho_{ti}^{(x)}\right] + \mathrm{ELTG}\left[\rho_{tj}^{(y)}\right]\right\} \cdot \sum_{r=1}^{d} \exp\left\{\mathrm{ELG}\left[x_{ir}\right] + \mathrm{ELG}\left[y_{jr}\right]\right\} \\
&\text{and } \boldsymbol{\chi}_{tij}^{\mathrm{new}} \overset{\text{def.}}{=} \left(\frac{\exp\left\{\mathrm{ELG}[x_{ir}] + \mathrm{ELG}[y_{jr}]\right\}}{\sum_{r=1}^{d} \exp\left\{\mathrm{ELG}[x_{ir}] + \mathrm{ELG}[y_{jr}]\right\}}\right)_{r=1}^{d}.
\end{aligned}
$$

3.5.2 Time-invariant latent features

Keeping in mind that $\log \mathrm{Gamma}\left(x\ ;\ \alpha, \beta\right) = \alpha \log \beta - \log \Gamma(\alpha) + (\alpha - 1)\log x - \beta x$, we have:

$$
\begin{aligned}
\log q^*(x_{ir}) &\propto \mathbb{E}_{q(\mathcal{P}\setminus\{x_{ir}\})}\left[\log p\left(x_{ir} \mid \mathcal{P} \setminus \{x_{ir}\}, \mathbf{A}\right)\right] \\
&\propto \log(x_{ir}) \cdot \left\{a^{(x)} - 1 + \sum_{t=1}^{T}\sum_{j=1}^{N_2} A_{tij} \cdot \frac{\phi_{tij}}{1 - \exp\{-\phi_{tij}\}} \cdot \chi_{tijr}\right\} \\
&\quad - x_{ir} \cdot \left\{\frac{\delta_i^{(x)}}{\eta_i^{(x)}} + \sum_{t=1}^{T}\sum_{j=1}^{N_2} \mathrm{ETG}\left[\rho_{ti}^{(x)}\right] \mathrm{ETG}\left[\rho_{tj}^{(y)}\right] \frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right\}.
\end{aligned}
$$

Since the support of $q^*(x_{ir})$ is given by the support of $p(x_{ir} \mid \mathcal{P} \setminus \{x_{ir}\}, \mathbf{A})$, the update is:

$$
q^*(x_{ir}) \sim \mathrm{Gamma}\left(a^{(x)} + \sum_{t,j:A_{tij}=1} \frac{\phi_{tij}}{1 - \exp\{-\phi_{tij}\}} \cdot \chi_{tijr}\ ,\ \frac{\delta_i^{(x)}}{\eta_i^{(x)}} + \sum_{t=1}^{T}\sum_{j=1}^{N_2} \mathrm{ETG}\left[\rho_{ti}^{(x)}\right] \mathrm{ETG}\left[\rho_{tj}^{(y)}\right] \frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right).
$$

By symmetry, the **y**-component update has the following form:

$$
q^*(y_{jr}) \sim \mathrm{Gamma}\left(a^{(y)} + \sum_{t,i:A_{tij}=1} \frac{\phi_{tij}}{1 - \exp\{-\phi_{tij}\}} \cdot \chi_{tijr}\ ,\ \frac{\delta_j^{(y)}}{\eta_j^{(y)}} + \sum_{t=1}^{T}\sum_{i=1}^{N_1} \mathrm{ETG}\left[\rho_{ti}^{(x)}\right] \mathrm{ETG}\left[\rho_{tj}^{(y)}\right] \frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}}\right).
$$

3.5.3 Spread Hyperparameters

Keeping in mind that $\log \mathrm{Gamma}\,(x\ ;\ \alpha, \beta) = \alpha \log \beta - \log \Gamma(\alpha) + (\alpha - 1)\log x - \beta x$, we have:

$$\begin{aligned}\log q^*(\zeta_i^{(x)}) &\propto \mathbb{E}_{q(\mathcal{P}\setminus\{\zeta_i^{(x)}\})}\left[\log p\left(\zeta_i^{(x)} \mid \mathcal{P}\setminus\{\zeta_i^{(x)}\}, \mathbf{A}\right)\right] \\ &\propto \log(\zeta_i^{(x)}) \cdot \left\{d \cdot a^{(x)} + b^{(x)} - 1\right\} - \zeta_i^{(x)} \cdot \left\{c^{(x)} + \sum_{r=1}^{d} \frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}}\right\}.\end{aligned}$$

Since the support of $q^*(\zeta_i^{(x)})$ is given by the support of $p(\zeta_i^{(x)} \mid \mathcal{P}\setminus\{\zeta_i^{(x)}\}, \mathbf{A})$, the update is:

$$q^*(\zeta_i^{(x)}) \sim \mathrm{Gamma}\left(d \cdot a^{(x)} + b^{(x)}\ ,\ c^{(x)} + \sum_{r=1}^{d} \frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}}\right).$$

By symmetry, the $\boldsymbol{\zeta}^{(y)}$-component update has the following form:

$$q^*(\zeta_j^{(y)}) \sim \mathrm{Gamma}\left(d \cdot a^{(y)} + b^{(y)}\ ,\ c^{(y)} + \sum_{r=1}^{d} \frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right).$$

3.5.4 Time-dependent correction factors

Keeping in mind that $\log \mathrm{Gamma}_{(0,1)}\,(x\ ;\ \alpha, \beta) = \alpha \log \beta - \log \gamma(\alpha, \beta) + (\alpha - 1)\log x - \beta x$, we have:

$$\begin{aligned}\log q^*(\rho_{ti}^{(x)}) &\propto \mathbb{E}_{q(\mathcal{P}\setminus\{\rho_{ti}^{(x)}\})}\left[\log p\left(\rho_{ti}^{(x)} \mid \mathcal{P}\setminus\{\rho_{ti}^{(x)}\}, \mathbf{A}\right)\right] \\ &\propto \log(\rho_{ti}^{(x)}) \cdot \left\{\alpha^{(x)} - 1 + \sum_{j=1}^{N_2} A_{tij} \cdot \frac{\phi_{tij}}{1 - \exp\{-\phi_{tij}\}}\right\} \\ &\quad - \rho_{ti}^{(x)} \cdot \left\{\beta^{(x)} + \sum_{j=1}^{N_2}\sum_{r=1}^{d} \mathrm{ETG}\left[\rho_{tj}^{(y)}\right] \cdot \frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}} \cdot \frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right\}.\end{aligned}$$

Since the support of $q^*(\rho_{ti}^{(x)})$ is given by the support of $p(\rho_{ti}^{(x)} \mid \mathcal{P}\setminus\{\rho_{ti}^{(x)}\}, \mathbf{A})$, the update is:

$$q^*(\rho_{ti}^{(x)}) \sim \mathrm{Gamma}_{(0,1)}\left(\alpha^{(x)} + \sum_{j:A_{tij}=1} \frac{\phi_{tij}}{1 - \exp\{-\phi_{tij}\}}\ ,\ \beta^{(x)} + \sum_{j=1}^{N_2}\sum_{r=1}^{d} \mathrm{ETG}\left[\rho_{tj}^{(y)}\right] \cdot \frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}} \cdot \frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right).$$

By symmetry, the $\boldsymbol{\rho}^{(y)}$-component update has the following form:

$$q^*(\rho_{tj}^{(y)}) \sim \mathrm{Gamma}_{(0,1)}\left(\alpha^{(y)} + \sum_{i:A_{tij}=1} \frac{\phi_{tij}}{1 - \exp\{-\phi_{tij}\}}\ ,\ \beta^{(y)} + \sum_{i=1}^{N_1}\sum_{r=1}^{d} \mathrm{ETG}\left[\rho_{ti}^{(x)}\right] \cdot \frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}} \cdot \frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right).$$

3.6 ELBO

The ELBO is defined as:

$$\mathrm{ELBO}(q \parallel p) \overset{\text{def.}}{=} \mathbb{E}_q\left[\log \frac{p\left(\boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}, \mathbf{x}, \mathbf{y}, \mathbf{Z}, \mathbf{N}, \mathbf{A}\right)}{q\left(\boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}, \boldsymbol{\zeta}^{(x)}, \boldsymbol{\zeta}^{(y)}, \mathbf{x}, \mathbf{y}, \mathbf{Z}, \mathbf{N}\right)}\right].$$

We can factorize the two densities as:

$$\begin{aligned}
&p\left(\boldsymbol{\rho}^{(x)},\boldsymbol{\rho}^{(y)},\boldsymbol{\zeta}^{(x)},\boldsymbol{\zeta}^{(y)},\mathbf{x},\mathbf{y},\mathbf{Z},\mathbf{N},\mathbf{A}\right)=p\left(\mathbf{A}\mid\mathbf{N}\right)p\left(\mathbf{N}\mid\mathbf{Z}\right)\times\\
&\qquad\times p\left(\mathbf{Z}\mid\mathbf{x},\mathbf{y},\boldsymbol{\rho}^{(x)},\boldsymbol{\rho}^{(y)}\right)p\left(\mathbf{x}\mid\boldsymbol{\zeta}^{(x)}\right)p\left(\mathbf{y}\mid\boldsymbol{\zeta}^{(y)}\right)p\left(\boldsymbol{\zeta}^{(x)}\right)p\left(\boldsymbol{\zeta}^{(y)}\right)p\left(\boldsymbol{\rho}^{(x)}\right)p\left(\boldsymbol{\rho}^{(y)}\right),\\
&q\left(\boldsymbol{\rho}^{(x)},\boldsymbol{\rho}^{(y)},\boldsymbol{\zeta}^{(x)},\boldsymbol{\zeta}^{(y)},\mathbf{x},\mathbf{y},\mathbf{Z},\mathbf{N}\right)\\
&\qquad=q\left(\mathbf{Z},\mathbf{N}\right)q\left(\mathbf{x}\right)q\left(\mathbf{y}\right)q\left(\boldsymbol{\zeta}^{(x)}\right)q\left(\boldsymbol{\zeta}^{(y)}\right)q\left(\boldsymbol{\rho}^{(x)}\right)q\left(\boldsymbol{\rho}^{(y)}\right).
\end{aligned}$$

Remember that, on the support of q, we have $A_{tij}=\mathbb{1}_{\mathbb{N}_+}(N_{tij})$ and $N_{tij}=\sum_{r=1}^{d}Z_{tijr}$ for every t,i,j. Hence, in the previous expectation, we may ignore the terms $p\left(\mathbf{A}\mid\mathbf{N}\right)$ and $p\left(\mathbf{N}\mid\mathbf{Z}\right)$.

Using these decompositions, the ELBO can be rewritten as:

$$\begin{aligned}
\mathrm{ELBO}(q\parallel p)=\mathbb{E}_q\left[\log\frac{p\left(\mathbf{Z}\mid\mathbf{x},\mathbf{y},\boldsymbol{\rho}^{(x)},\boldsymbol{\rho}^{(y)}\right)}{q\left(\mathbf{Z},\mathbf{N}\right)}\right]+\mathbb{E}_q\left[\log\frac{p\left(\mathbf{x}\mid\boldsymbol{\zeta}^{(x)}\right)p\left(\mathbf{y}\mid\boldsymbol{\zeta}^{(y)}\right)}{q\left(\mathbf{x}\right)q\left(\mathbf{y}\right)}\right]\\
+\mathbb{E}_q\left[\log\frac{p\left(\boldsymbol{\zeta}^{(x)}\right)p\left(\boldsymbol{\zeta}^{(y)}\right)}{q\left(\boldsymbol{\zeta}^{(x)}\right)q\left(\boldsymbol{\zeta}^{(y)}\right)}\right]+\mathbb{E}_q\left[\log\frac{p\left(\boldsymbol{\rho}^{(x)}\right)p\left(\boldsymbol{\rho}^{(y)}\right)}{q\left(\boldsymbol{\rho}^{(x)}\right)q\left(\boldsymbol{\rho}^{(y)}\right)}\right].
\end{aligned}$$

We identify four components: the likelihood component, the time-invariant latent feature component, the time-dependent correction factor component, and the spread hyperparameter component.

3.6.1 Likelihood component

The likelihood component of the ELBO is:

$$\begin{aligned}
\mathbb{E}_q\left[\log\frac{p\left(\mathbf{Z}\mid\mathbf{x},\mathbf{y},\boldsymbol{\rho}^{(x)},\boldsymbol{\rho}^{(y)}\right)}{q\left(\mathbf{Z},\mathbf{N}\right)}\right]&=\mathbb{E}_q\left[\log p\left(\mathbf{Z}\mid\mathbf{x},\mathbf{y},\boldsymbol{\rho}^{(x)},\boldsymbol{\rho}^{(y)}\right)-\log q\left(\mathbf{Z},\mathbf{N}\right)\right]\\
=\mathbb{E}_q\left[\log p\left(\mathbf{Z}\mid\mathbf{x},\mathbf{y},\boldsymbol{\rho}^{(x)},\boldsymbol{\rho}^{(y)}\right)+\sum_{t,i,j,r}\log(Z_{tijr}!)\right]&-\mathbb{E}_q\left[\log q\left(\mathbf{Z},\mathbf{N}\right)+\sum_{t,i,j,r}\log(Z_{tijr}!)\right].
\end{aligned}$$

Keeping in mind that $q(\mathbf{Z}_{tij},N_{tij})=A_{tij}\times\mathrm{Poisson}_+\left(N_{tij}\ ;\ \phi_{tij}\right)\times\mathrm{Multinomial}\left(\mathbf{Z}_{tij}\ ;\ N_{tij},\boldsymbol{\chi}_{tij}\right)$, $p(Z_{tijr}\mid x_{ir},y_{jr},\rho_{ti}^{(x)},\rho_{tj}^{(y)})\sim\mathrm{Poisson}(\rho_{ti}^{(x)}\rho_{tj}^{(y)}x_{ir}y_{jr})$, and $\log\mathrm{Poisson}(x;\lambda)=x\log\lambda-\lambda-\log(x!)$, we have:

$$\begin{aligned}
\log p\Big(\mathbf{Z}\mid\mathbf{x},\mathbf{y},\boldsymbol{\rho}^{(x)},\boldsymbol{\rho}^{(y)}\Big)&=\sum_{t=1}^{T}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{r=1}^{d}\Big\{Z_{tijr}\log\left(\rho_{ti}^{(x)}\rho_{tj}^{(y)}x_{ir}y_{jr}\right)-\rho_{ti}^{(x)}\rho_{tj}^{(y)}x_{ir}y_{jr}-\log(Z_{tijr}!)\Big\},\\
\log q\Big(\mathbf{Z},\mathbf{N}\Big)&=\sum_{t,i,j:A_{tij}=1}\log\left[\frac{\exp(-\phi_{tij})}{1-\exp(-\phi_{tij})}\frac{\phi_{tij}^{N_{tij}}}{N_{tij}!}\cdot\frac{N_{tij}!}{\prod_{r=1}^{d}Z_{tijr}!}\prod_{r=1}^{d}\chi_{tijr}^{Z_{tijr}}\right]\\
&=\sum_{t,i,j:A_{tij}=1}\left\{N_{tij}\log(\phi_{tij})-\phi_{tij}-\log\left(1-\exp(-\phi_{tij})\right)-\sum_{r=1}^{d}\log(Z_{tijr}!)+\sum_{r=1}^{d}Z_{tijr}\log(\chi_{tijr})\right\}.
\end{aligned}$$

Notice that $Z_{tijr}=0$ for all $r=1,\cdots,d$ whenever $A_{tij}=0$. Taking expectations, we get:

$$\begin{aligned}
&\mathbb{E}_q\left[\log p\left(\mathbf{Z}\mid\mathbf{x},\mathbf{y},\boldsymbol{\rho}^{(x)},\boldsymbol{\rho}^{(y)}\right)+\sum_{t,i,j,r}\log(Z_{tijr}!)\right]\\
&=\sum_{t,i,j:A_{tij}=1}\sum_{r=1}^{d}\frac{\phi_{tij}}{1-\exp\{-\phi_{tij}\}}\chi_{tijr}\cdot\Big\{\mathrm{ELG}\left[x_{ir}\right]+\mathrm{ELG}\left[y_{jr}\right]+\mathrm{ELTG}\left[\rho_{ti}^{(x)}\right]+\mathrm{ELTG}\left[\rho_{tj}^{(y)}\right]\Big\}
\end{aligned}$$

$$
- \sum_{t=1}^{T} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sum_{r=1}^{d} \mathrm{ETG}\left[\rho_{ti}^{(x)}\right] \mathrm{ETG}\left[\rho_{tj}^{(y)}\right] \frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}} \frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}},
$$

$$
\begin{aligned}
&\mathbb{E}_q\left[\log q(\mathbf{Z},\mathbf{N}) + \sum_{t,i,j,r} \log(Z_{tijr}!)\right] = \\
&= \sum_{t,i,j:A_{tij}=1} \left\{ \frac{\phi_{tij}\cdot\log(\phi_{tij})}{1-\exp(-\phi_{tij})} - \phi_{tij} - \log\left(1-\exp(-\phi_{tij})\right) + \frac{\phi_{tij}}{1-\exp\{-\phi_{tij}\}} \sum_{r=1}^{d} \chi_{tijr}\log(\chi_{tijr}) \right\} \\
&= \sum_{t,i,j:A_{tij}=1} \left\{ \frac{\phi_{tij}}{1-\exp(-\phi_{tij})}\log(\phi_{tij}) - \log\left(\exp(\phi_{tij})-1\right) + \frac{\phi_{tij}}{1-\exp\{-\phi_{tij}\}} \sum_{r=1}^{d} \chi_{tijr}\log(\chi_{tijr}) \right\}.
\end{aligned}
$$

since $\phi_{tij} = \log\left(\exp(\phi_{tij})\right)$.

We get the following expression for the likelihood component of the ELBO:

$$
\begin{aligned}
&\mathbb{E}_q\left[\log \frac{p\left(\mathbf{Z}\mid\mathbf{x},\mathbf{y},\boldsymbol{\rho}^{(x)},\boldsymbol{\rho}^{(y)}\right)}{q(\mathbf{Z},\mathbf{N})}\right] \\
&\quad= \mathbb{E}_q\left[\log p\left(\mathbf{Z}\mid\mathbf{x},\mathbf{y},\boldsymbol{\rho}^{(x)},\boldsymbol{\rho}^{(y)}\right) + \sum_{t,i,j,r}\log(Z_{tijr}!)\right] - \mathbb{E}_q\left[\log q(\mathbf{Z},\mathbf{N}) + \sum_{t,i,j,r}\log(Z_{tijr}!)\right] \\
&\quad= \sum_{t,i,j:A_{tij}=1} \log\left(\exp(\phi_{tij})-1\right) - \sum_{t=1}^{T}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{r=1}^{d} \mathrm{ETG}\left[\rho_{ti}^{(x)}\right]\mathrm{ETG}\left[\rho_{tj}^{(y)}\right]\frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}}\frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}} \\
&\qquad+ \sum_{t,i,j:A_{tij}=1} \frac{\phi_{tij}}{1-\exp(-\phi_{tij})}\Bigg\{ -\log(\phi_{tij}) + \\
&\qquad\qquad+ \sum_{r=1}^{d}\chi_{tijr}\cdot\left[\mathrm{ELG}[x_{ir}] + \mathrm{ELG}[y_{jr}] + \mathrm{ELTG}\left[\rho_{ti}^{(x)}\right] + \mathrm{ELTG}\left[\rho_{tj}^{(y)}\right] - \log(\chi_{tijr})\right]\Bigg\}.
\end{aligned}
$$

Terms such as $\log(\phi_{tij})$ and $\log(\chi_{tijr})$ might be numerically unstable if ϕ_{tij} or χ_{tijr} are close to 0. In order to avoid this issue, we are going to make an assumption that can further simplify the ELBO.

Assuming that the ELBO is computed just after the Z-N component update, we can simplify it further. Remember that the update is given by:

$$
\begin{aligned}
\phi_{tij}^{\text{new}} &\overset{\text{def.}}{=} \exp\left\{\mathrm{ELTG}\left[\rho_{ti}^{(x)}\right] + \mathrm{ELTG}\left[\rho_{tj}^{(y)}\right]\right\}\cdot\sum_{r=1}^{d}\exp\left\{\mathrm{ELG}[x_{ir}] + \mathrm{ELG}[y_{jr}]\right\}, \\
\boldsymbol{\chi}_{tij}^{\text{new}} &\overset{\text{def.}}{=} \left(\frac{\exp\{\mathrm{ELG}[x_{ir}] + \mathrm{ELG}[y_{jr}]\}}{\sum_{r=1}^{d}\exp\{\mathrm{ELG}[x_{ir}] + \mathrm{ELG}[y_{jr}]\}}\right)_{r=1}^{d}.
\end{aligned}
$$

These expressions can be used to simplify the ELBO:

$$
\begin{aligned}
\mathrm{ELG}[x_{ir}] + \mathrm{ELG}[y_{jr}] &= \log(\chi_{tijr}) + \log\left\{\sum_{r=1}^{d}\exp\{\mathrm{ELG}[x_{ir}] + \mathrm{ELG}[y_{jr}]\}\right\}, \\
\mathrm{ELTG}\left[\rho_{ti}^{(x)}\right] + \mathrm{ELTG}\left[\rho_{tj}^{(y)}\right] &= \log(\phi_{tij}) - \log\left\{\sum_{r=1}^{d}\exp\{\mathrm{ELG}[x_{ir}] + \mathrm{ELG}[y_{jr}]\}\right\}, \\
\sum_{r=1}^{d}\chi_{tijr}\cdot\log(\phi_{tij}) &= \log(\phi_{tij}) \text{ since } \sum_{r=1}^{d}\chi_{tijr} = 1.
\end{aligned}
$$

Hence, the third sum reduces to 0. We get the following expression for the likelihood component of the ELBO, *in the case where the ELBO is computed just after the Z-N component update*:

$$\begin{aligned}
&\mathbb{E}_q\left[\log \frac{p\left(\mathbf{Z} \mid \mathbf{x}, \mathbf{y}, \boldsymbol{\rho}^{(x)}, \boldsymbol{\rho}^{(y)}\right)}{q\left(\mathbf{Z}, \mathbf{N}\right)}\right] \\
&\quad = \sum_{t,i,j: A_{tij}=1} \log\left(\exp(\phi_{tij}) - 1\right) - \sum_{t=1}^{T}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{r=1}^{d} \mathrm{ETG}\left[\rho_{ti}^{(x)}\right] \mathrm{ETG}\left[\rho_{tj}^{(y)}\right] \frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}} \frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}.
\end{aligned}$$

Note: In order to reduce the time complexity of the second sum, we are going to use the following trick:

$$\sum_{t=1}^{T}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{r=1}^{d} \mathrm{ETG}\left[\rho_{ti}^{(x)}\right] \mathrm{ETG}\left[\rho_{tj}^{(y)}\right] \frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}} \frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}} = \sum_{t=1}^{T}\sum_{r=1}^{d} \left(\sum_{i=1}^{N_1} \mathrm{ETG}\left[\rho_{ti}^{(x)}\right] \frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}}\right) \left(\sum_{j=1}^{N_2} \mathrm{ETG}\left[\rho_{tj}^{(y)}\right] \frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right).$$

3.6.2 Time-invariant latent feature component

The time-invariant latent feature component of the ELBO is:

$$\begin{aligned}
\mathbb{E}_q\left[\log \frac{p\left(\mathbf{x} \mid \boldsymbol{\zeta}^{(x)}\right) p\left(\mathbf{y} \mid \boldsymbol{\zeta}^{(y)}\right)}{q\left(\mathbf{x}\right) q\left(\mathbf{y}\right)}\right] &= \mathbb{E}_q\left[\log \frac{p\left(\mathbf{x} \mid \boldsymbol{\zeta}^{(x)}\right)}{q\left(\mathbf{x}\right)}\right] + \mathbb{E}_q\left[\log \frac{p\left(\mathbf{y} \mid \boldsymbol{\zeta}^{(y)}\right)}{q\left(\mathbf{y}\right)}\right] \\
&= \mathbb{E}_q\left[\log p\left(\mathbf{x} \mid \boldsymbol{\zeta}^{(x)}\right)\right] + \mathbb{E}_q\left[\log p\left(\mathbf{y} \mid \boldsymbol{\zeta}^{(y)}\right)\right] - \mathbb{E}_q\left[\log q\left(\mathbf{x}\right)\right] - \mathbb{E}_q\left[\log q\left(\mathbf{y}\right)\right].
\end{aligned}$$

Keeping in mind that $\log \mathrm{Gamma}\left(x\ ;\ \alpha, \beta\right) = \alpha \log \beta - \log \Gamma(\alpha) + (\alpha - 1)\log x - \beta x$, $p(x_{ir} \mid \zeta_i^{(x)}) \sim \mathrm{Gamma}(a^{(x)}, \zeta_i^{(x)})$, and $q(x_{ir}) \sim \mathrm{Gamma}\left(\lambda_{ir}^{(x)}, \mu_{ir}^{(x)}\right)$, and similarly for y_{jr}, we have:

$$\begin{aligned}
\log p\left(\mathbf{x} \mid \boldsymbol{\zeta}^{(x)}\right) &= \sum_{i=1}^{N_1}\sum_{r=1}^{d} \left\{a^{(x)} \log\left(\zeta_i^{(x)}\right) - \log\Gamma(a^{(x)}) + (a^{(x)} - 1)\log\left(x_{ir}\right) - \zeta_i^{(x)} x_{ir}\right\} \\
&= (-N_1 d)\log\Gamma(a^{(x)}) + d a^{(x)} \sum_{i=1}^{N_1} \log\left(\zeta_i^{(x)}\right) + \sum_{i=1}^{N_1}\sum_{r=1}^{d} \left\{(a^{(x)} - 1)\log\left(x_{ir}\right) - \zeta_i^{(x)} x_{ir}\right\}, \\
\log p\left(\mathbf{y} \mid \boldsymbol{\zeta}^{(y)}\right) &= (-N_2 d)\log\Gamma(a^{(y)}) + d a^{(y)} \sum_{j=1}^{N_2} \log\left(\zeta_j^{(y)}\right) + \sum_{j=1}^{N_2}\sum_{r=1}^{d} \left\{(a^{(y)} - 1)\log\left(y_{jr}\right) - \zeta_j^{(y)} y_{jr}\right\}, \\
\log q\left(\mathbf{x}\right) &= \sum_{i=1}^{N_1}\sum_{r=1}^{d} \left\{\lambda_{ir}^{(x)} \log(\mu_{ir}^{(x)}) - \log\Gamma(\lambda_{ir}^{(x)}) + (\lambda_{ir}^{(x)} - 1)\log(x_{ir}) - \mu_{ir}^{(x)} x_{ir}\right\}, \\
\log q\left(\mathbf{y}\right) &= \sum_{j=1}^{N_2}\sum_{r=1}^{d} \left\{\lambda_{jr}^{(y)} \log(\mu_{jr}^{(y)}) - \log\Gamma(\lambda_{jr}^{(y)}) + (\lambda_{jr}^{(y)} - 1)\log(y_{jr}) - \mu_{jr}^{(y)} y_{jr}\right\}.
\end{aligned}$$

Taking expectations, we get:

$$\begin{aligned}
\mathbb{E}_q\left[\log p\left(\mathbf{x} \mid \boldsymbol{\zeta}^{(x)}\right)\right] &= -N_1 d \log\Gamma(a^{(x)}) + d a^{(x)} \sum_{i=1}^{N_1} \mathrm{ELG}\left[\zeta_i^{(x)}\right] + \sum_{i=1}^{N_1}\sum_{r=1}^{d} \left\{(a^{(x)} - 1)\mathrm{ELG}\left[x_{ir}\right] - \frac{\delta_i^{(x)}}{\eta_i^{(x)}} \frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}}\right\}, \\
\mathbb{E}_q\left[\log p\left(\mathbf{y} \mid \boldsymbol{\zeta}^{(y)}\right)\right] &= -N_2 d \log\Gamma(a^{(y)}) + d a^{(y)} \sum_{j=1}^{N_2} \mathrm{ELG}\left[\zeta_j^{(y)}\right] + \sum_{j=1}^{N_2}\sum_{r=1}^{d} \left\{(a^{(y)} - 1)\mathrm{ELG}\left[y_{jr}\right] - \frac{\delta_j^{(y)}}{\eta_j^{(y)}} \frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right\}, \\
\mathbb{E}_q\left[\log q\left(\mathbf{x}\right)\right] &= \sum_{i=1}^{N_1}\sum_{r=1}^{d} \left\{\lambda_{ir}^{(x)} \log(\mu_{ir}^{(x)}) - \log\Gamma(\lambda_{ir}^{(x)}) + (\lambda_{ir}^{(x)} - 1)\mathrm{ELG}\left[x_{ir}\right] - \lambda_{ir}^{(x)}\right\},
\end{aligned}$$

$$\mathbb{E}_q\left[\log q\left(\mathbf{y}\right)\right] = \sum_{j=1}^{N_2}\left\{\sum_{r=1}^{d}\lambda_{jr}^{(y)}\log(\mu_{jr}^{(y)}) - \log\Gamma(\lambda_{jr}^{(y)}) + (\lambda_{jr}^{(y)}-1)\mathrm{ELG}\left[y_{jr}\right] - \lambda_{jr}^{(y)}\right\}.$$

3.6.3 Spread Hyperparameter component

The spread hyperparameter component is:

$$\begin{aligned}\mathbb{E}_q\left[\log\frac{p\left(\boldsymbol{\zeta}^{(x)}\right)p\left(\boldsymbol{\zeta}^{(y)}\right)}{q\left(\boldsymbol{\zeta}^{(x)}\right)q\left(\boldsymbol{\zeta}^{(y)}\right)}\right] &= \mathbb{E}_q\left[\log\frac{p\left(\boldsymbol{\zeta}^{(x)}\right)}{q\left(\boldsymbol{\zeta}^{(x)}\right)}\right] + \mathbb{E}_q\left[\log\frac{p\left(\boldsymbol{\zeta}^{(y)}\right)}{q\left(\boldsymbol{\zeta}^{(y)}\right)}\right]\\ &= \mathbb{E}_q\left[\log p\left(\boldsymbol{\zeta}^{(x)}\right)\right] + \mathbb{E}_q\left[\log p\left(\boldsymbol{\zeta}^{(y)}\right)\right] - \mathbb{E}_q\left[\log q\left(\boldsymbol{\zeta}^{(x)}\right)\right] - \mathbb{E}_q\left[\log q\left(\boldsymbol{\zeta}^{(y)}\right)\right].\end{aligned}$$

Keeping in mind that $\log\mathrm{Gamma}\left(x\ ;\ \alpha,\beta\right) = \alpha\log\beta - \log\Gamma(\alpha) + (\alpha-1)\log x - \beta x$, $p(\zeta_i^{(x)}) \sim \mathrm{Gamma}(b^{(x)}, c^{(x)})$, $q(\zeta_i^{(x)}) \sim \mathrm{Gamma}\left(\delta_i^{(x)}, \eta_i^{(x)}\right)$, and similarly for $\zeta_j^{(y)}$, we have:

$$\begin{aligned}\log p\left(\boldsymbol{\zeta}^{(x)}\right) &= \sum_{i=1}^{N_1}\left\{b^{(x)}\log(c^{(x)}) - \log\Gamma(b^{(x)}) + (b^{(x)}-1)\log\left(\zeta_i^{(x)}\right) - c^{(x)}\zeta_i^{(x)}\right\}\\ &= N_1 b^{(x)}\log(c^{(x)}) - N_1\log\Gamma(b^{(x)}) + \sum_{i=1}^{N_1}\left\{(b^{(x)}-1)\log\left(\zeta_i^{(x)}\right) - c^{(x)}\zeta_i^{(x)}\right\},\\ \log p\left(\boldsymbol{\zeta}^{(y)}\right) &= N_2 b^{(y)}\log(c^{(y)}) - N_2\log\Gamma(b^{(y)}) + \sum_{j=1}^{N_2}\left\{(b^{(y)}-1)\log\left(\zeta_j^{(y)}\right) - c^{(y)}\zeta_j^{(y)}\right\},\\ \log q\left(\boldsymbol{\zeta}^{(x)}\right) &= \sum_{i=1}^{N_1}\left\{\delta_i^{(x)}\log(\eta_i^{(x)}) - \log\Gamma(\delta_i^{(x)}) + (\delta_i^{(x)}-1)\log\left(\zeta_i^{(x)}\right) - \eta_i^{(x)}\zeta_i^{(x)}\right\},\\ \log q\left(\boldsymbol{\zeta}^{(y)}\right) &= \sum_{j=1}^{N_2}\left\{\delta_j^{(y)}\log(\eta_j^{(y)}) - \log\Gamma(\delta_j^{(y)}) + (\delta_j^{(y)}-1)\log\left(\zeta_j^{(y)}\right) - \eta_j^{(y)}\zeta_j^{(y)}\right\}.\end{aligned}$$

Taking expectations, we get:

$$\begin{aligned}\mathbb{E}_q\left[\log p\left(\boldsymbol{\zeta}^{(x)}\right)\right] &= N_1 b^{(x)}\log(c^{(x)}) - N_1\log\Gamma(b^{(x)}) + \sum_{i=1}^{N_1}\left\{(b^{(x)}-1)\mathrm{ELG}\left[\zeta_i^{(x)}\right] - c^{(x)}\frac{\delta_i^{(x)}}{\eta_i^{(x)}}\right\},\\ \mathbb{E}_q\left[\log p\left(\boldsymbol{\zeta}^{(y)}\right)\right] &= N_2 b^{(y)}\log(c^{(y)}) - N_2\log\Gamma(b^{(y)}) + \sum_{j=1}^{N_2}\left\{(b^{(y)}-1)\mathrm{ELG}\left[\zeta_j^{(y)}\right] - c^{(y)}\frac{\delta_j^{(y)}}{\eta_j^{(y)}}\right\},\\ \mathbb{E}_q\left[\log q\left(\boldsymbol{\zeta}^{(x)}\right)\right] &= \sum_{i=1}^{N_1}\left\{\delta_i^{(x)}\log(\eta_i^{(x)}) - \log\Gamma(\delta_i^{(x)}) + (\delta_i^{(x)}-1)\mathrm{ELG}\left[\zeta_i^{(x)}\right] - \delta_i^{(x)}\right\},\\ \mathbb{E}_q\left[\log q\left(\boldsymbol{\zeta}^{(y)}\right)\right] &= \sum_{j=1}^{N_2}\left\{\delta_j^{(y)}\log(\eta_j^{(y)}) - \log\Gamma(\delta_j^{(y)}) + (\delta_j^{(y)}-1)\mathrm{ELG}\left[\zeta_j^{(y)}\right] - \delta_j^{(y)}\right\}.\end{aligned}$$

3.6.4 Time-dependent correction factor component

The time-dependent correction factor component is:

$$\mathbb{E}_q\left[\log\frac{p\left(\boldsymbol{\rho}^{(x)}\right)p\left(\boldsymbol{\rho}^{(y)}\right)}{q\left(\boldsymbol{\rho}^{(x)}\right)q\left(\boldsymbol{\rho}^{(y)}\right)}\right] = \mathbb{E}_q\left[\log\frac{p\left(\boldsymbol{\rho}^{(x)}\right)}{q\left(\boldsymbol{\rho}^{(x)}\right)}\right] + \mathbb{E}_q\left[\log\frac{p\left(\boldsymbol{\rho}^{(y)}\right)}{q\left(\boldsymbol{\rho}^{(y)}\right)}\right]$$

$$= \mathbb{E}_q\left[\log p\left(\boldsymbol{\rho}^{(x)}\right)\right] + \mathbb{E}_q\left[\log p\left(\boldsymbol{\rho}^{(y)}\right)\right] - \mathbb{E}_q\left[\log q\left(\boldsymbol{\rho}^{(x)}\right)\right] - \mathbb{E}_q\left[\log q\left(\boldsymbol{\rho}^{(y)}\right)\right].$$

Keeping in mind that $\log \mathrm{Gamma}_{(0,1)}\left(x\ ;\ \alpha, \beta\right) = \alpha \log \beta - \log \gamma(\alpha,\beta) + (\alpha-1)\log x - \beta x$, $p(\rho_{ti}^{(x)}) \sim \mathrm{Gamma}_{(0,1)}(\alpha^{(x)}, \beta^{(x)})$, $q(\rho_{ti}^{(x)}) \sim \mathrm{Gamma}_{(0,1)}(m_{ti}^{(x)}, n_{ti}^{(x)})$, and similarly for $\rho_{tj}^{(y)}$, we have:

$$\begin{aligned}
\log p\left(\boldsymbol{\rho}^{(x)}\right) &= \sum_{t=1}^{T}\sum_{i=1}^{N_1}\left\{\alpha^{(x)}\log(\beta^{(x)}) - \log\gamma(\alpha^{(x)},\beta^{(x)}) + (\alpha^{(x)}-1)\log(\rho_{ti}^{(x)}) - \beta^{(x)}\rho_{ti}^{(x)}\right\} \\
&= (TN_1)\alpha^{(x)}\log(\beta^{(x)}) - (TN_1)\log\gamma(\alpha^{(x)},\beta^{(x)}) + \sum_{t=1}^{T}\sum_{i=1}^{N_1}\left\{(\alpha^{(x)}-1)\log(\rho_{ti}^{(x)}) - \beta^{(x)}\rho_{ti}^{(x)}\right\}, \\
\log p\left(\boldsymbol{\rho}^{(y)}\right) &= (TN_2)\alpha^{(y)}\log(\beta^{(y)}) - (TN_2)\log\gamma(\alpha^{(y)},\beta^{(y)}) + \sum_{t=1}^{T}\sum_{j=1}^{N_2}\left\{(\alpha^{(y)}-1)\log(\rho_{tj}^{(y)}) - \beta^{(y)}\rho_{tj}^{(y)}\right\}, \\
\log q\left(\boldsymbol{\rho}^{(x)}\right) &= \sum_{t=1}^{T}\sum_{i=1}^{N_1}\left\{m_{ti}^{(x)}\log(n_{ti}^{(x)}) - \log\gamma(m_{ti}^{(x)},n_{ti}^{(x)}) + (m_{ti}^{(x)}-1)\log(\rho_{ti}^{(x)}) - n_{ti}^{(x)}\rho_{ti}^{(x)}\right\}, \\
\log q\left(\boldsymbol{\rho}^{(y)}\right) &= \sum_{t=1}^{T}\sum_{j=1}^{N_2}\left\{m_{tj}^{(y)}\log(n_{tj}^{(y)}) - \log\gamma(m_{tj}^{(y)},n_{tj}^{(y)}) + (m_{tj}^{(y)}-1)\log(\rho_{tj}^{(y)}) - n_{tj}^{(y)}\rho_{tj}^{(y)}\right\}.
\end{aligned}$$

Taking expectations, we get:

$$\begin{aligned}
\mathbb{E}_q\left[\log p\left(\boldsymbol{\rho}^{(x)}\right)\right] &= (TN_1)\alpha^{(x)}\log(\beta^{(x)}) - (TN_1)\log\gamma(\alpha^{(x)},\beta^{(x)}) \\
&\qquad + \sum_{t=1}^{T}\sum_{i=1}^{N_1}\left\{(\alpha^{(x)}-1)\mathrm{ELTG}\left[\rho_{ti}^{(x)}\right] - \beta^{(x)}\mathrm{ETG}\left[\rho_{ti}^{(x)}\right]\right\}, \\
\mathbb{E}_q\left[\log p\left(\boldsymbol{\rho}^{(y)}\right)\right] &= (TN_2)\alpha^{(y)}\log(\beta^{(y)}) - (TN_2)\log\gamma(\alpha^{(y)},\beta^{(y)}) \\
&\qquad + \sum_{t=1}^{T}\sum_{j=1}^{N_2}\left\{(\alpha^{(y)}-1)\mathrm{ELTG}\left[\rho_{tj}^{(y)}\right] - \beta^{(y)}\mathrm{ETG}\left[\rho_{tj}^{(y)}\right]\right\}, \\
\mathbb{E}_q\left[\log q\left(\boldsymbol{\rho}^{(x)}\right)\right] &= \sum_{t=1}^{T}\sum_{i=1}^{N_1}\left\{m_{ti}^{(x)}\log(n_{ti}^{(x)}) - \log\gamma(m_{ti}^{(x)},n_{ti}^{(x)}) + (m_{ti}^{(x)}-1)\mathrm{ELTG}\left[\rho_{ti}^{(x)}\right] - n_{ti}^{(x)}\mathrm{ETG}\left[\rho_{ti}^{(x)}\right]\right\}, \\
\mathbb{E}_q\left[\log q\left(\boldsymbol{\rho}^{(y)}\right)\right] &= \sum_{t=1}^{T}\sum_{j=1}^{N_2}\left\{m_{tj}^{(y)}\log(n_{tj}^{(y)}) - \log\gamma(m_{tj}^{(y)},n_{tj}^{(y)}) + (m_{tj}^{(y)}-1)\mathrm{ELTG}\left[\rho_{tj}^{(y)}\right] - n_{tj}^{(y)}\mathrm{ETG}\left[\rho_{tj}^{(y)}\right]\right\}.
\end{aligned}$$

3.7 Algorithm

Algorithm 3 summarizes the steps to perform CAVI in our model.

Remember that the likelihood component of the full ELBO contains terms such as $\log(\phi_{tij})$ and $\log(\chi_{tijr})$, which might be numerically unstable if ϕ_{tij} or χ_{tijr} are close to 0. By updating **Z** and **N** at the end, these unstable terms are eliminated from the ELBO, making the computation more stable.

We will then discuss the initialization of the parameters and the convergence criteria.

```
Algorithm 3: CAVI in our Model
  Input: A data set $\{\mathbf{A}_t\}$.
  Output: A mean-field variational density for our model.
  Initialize: The mean-field variational factors of our model.
  while the ELBO has not converged do
      Step 1: Update the distributions of $\mathbf{x}$ and $\mathbf{y}$:
        $$q^*(x_{ir}) \sim \text{Gamma}\left(a^{(x)} + \sum_{t,j:A_{tij}=1} \frac{\phi_{tij}\cdot\chi_{tijr}}{1-e^{-\phi_{tij}}}\ ,\ \frac{\delta_i^{(x)}}{\eta_i^{(x)}} + \sum_{t=1}^{T}\sum_{j=1}^{N_2} \text{ETG}\left[\rho_{ti}^{(x)}\right]\text{ETG}\left[\rho_{tj}^{(y)}\right]\frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right),$$
        $$q^*(y_{jr}) \sim \text{Gamma}\left(a^{(y)} + \sum_{t,i:A_{tij}=1} \frac{\phi_{tij}\cdot\chi_{tijr}}{1-e^{-\phi_{tij}}}\ ,\ \frac{\delta_j^{(y)}}{\eta_j^{(y)}} + \sum_{t=1}^{T}\sum_{i=1}^{N_1} \text{ETG}\left[\rho_{ti}^{(x)}\right]\text{ETG}\left[\rho_{tj}^{(y)}\right]\frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}}\right).$$
      Step 2: Update the distribution of $\boldsymbol{\zeta}^{(x)}$ and $\boldsymbol{\zeta}^{(y)}$:
        $$q^*(\zeta_i^{(x)}) \sim \text{Gamma}\left(d\cdot a^{(x)} + b^{(x)}\ ,\ c^{(x)} + \sum_{r=1}^{d}\frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}}\right),$$
        $$q^*(\zeta_j^{(y)}) \sim \text{Gamma}\left(d\cdot a^{(y)} + b^{(y)}\ ,\ c^{(y)} + \sum_{r=1}^{d}\frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right).$$
      Step 3: Update the distributions of $\boldsymbol{\rho}^{(x)}$ and $\boldsymbol{\rho}^{(y)}$:
        $$q^*(\rho_{ti}^{(x)}) \sim \text{Gamma}_{(0,1)}\left(\alpha^{(x)} + \sum_{j:A_{tij}=1}\frac{\phi_{tij}}{1-e^{-\phi_{tij}}}\ ,\ \beta^{(x)} + \sum_{j=1}^{N_2}\sum_{r=1}^{d}\text{ETG}\left[\rho_{tj}^{(y)}\right]\cdot\frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}}\cdot\frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right),$$
        $$q^*(\rho_{tj}^{(y)}) \sim \text{Gamma}_{(0,1)}\left(\alpha^{(y)} + \sum_{i:A_{tij}=1}\frac{\phi_{tij}}{1-e^{-\phi_{tij}}}\ ,\ \beta^{(y)} + \sum_{i=1}^{N_1}\sum_{r=1}^{d}\text{ETG}\left[\rho_{ti}^{(x)}\right]\cdot\frac{\lambda_{ir}^{(x)}}{\mu_{ir}^{(x)}}\cdot\frac{\lambda_{jr}^{(y)}}{\mu_{jr}^{(y)}}\right).$$
      Step 4: Update the joint distribution of the $\mathbf{Z}$ and $\mathbf{N}$:
        $$\begin{aligned} q^*(\mathbf{Z}_{tij}, N_{tij}) &= A_{tij}\times\text{Poisson}_+\left(N_{tij}\ ;\ \phi_{tij}^{\text{new}}\right)\times\text{Multinomial}\left(\mathbf{Z}_{tij}\ ;\ N_{tij}, \boldsymbol{\chi}_{tij}^{\text{new}}\right) \\ &\sim \begin{cases}\text{Poisson}_+\left(N_{tij}\ ;\ \phi_{tij}^{\text{new}}\right)\times\text{Multinomial}\left(\mathbf{Z}_{tij}\ ;\ N_{tij}, \boldsymbol{\chi}_{tij}^{\text{new}}\right) & \text{if } A_{tij}=1,\\ \delta_0\left(\mathbf{Z}_{tij}, N_{tij}\right) & \text{if } A_{tij}=0.\end{cases},\end{aligned}$$
        where $\phi_{tij}^{\text{new}} \overset{\text{def.}}{=} \exp\left\{\text{ELTG}\left[\rho_{ti}^{(x)}\right] + \text{ELTG}\left[\rho_{tj}^{(y)}\right]\right\}\cdot\sum_{r=1}^{d}\exp\left\{\text{ELG}\left[x_{ir}\right] + \text{ELG}\left[y_{jr}\right]\right\}$
        and $\boldsymbol{\chi}_{tij}^{\text{new}} \overset{\text{def.}}{=} \left(\frac{\exp\left\{\text{ELG}[x_{ir}] + \text{ELG}[y_{jr}]\right\}}{\sum_{r=1}^{d}\exp\left\{\text{ELG}[x_{ir}] + \text{ELG}[y_{jr}]\right\}}\right)_{r=1}^{d}.$
      Step 5: Compute the ELBO.
  end
  return The mean-variational family for our model.
```

3.7.1 Initialization

Step 1: Initialize the parameters for the count components $\mathbf{Z}$ and $\mathbf{N}$.

When $A_{tij} = 0$, we know that $Z_{tijr} = 0$ and $N_{tij} = 0$. There is no parameter to initialize in this case.

When $A_{tij} = 1$, we need to initialize ϕ_{tij} and $\boldsymbol{\chi}_{tij}$. Since we do not have any information about the latent features and the correction factors, we just initialize ϕ_{tij} to 1 and χ_{tijr} to $1/d$ (since r goes from 1 to d).

Step 2: Initialize the parameters for the latent features $\mathbf{x}$ and $\mathbf{y}$.

By using truncated SVD, we get a latent position encoding that might be close to the true latent positions. Let $\bar{\mathbf{A}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{A}_t$ be the average interaction matrix. Using truncated SVD with d singular values, we approximate $\bar{\mathbf{A}} \approx UDV^T$, where D is a diagonal matrix with the largest d singular values, and U and V are the corresponding left and right singular vectors, respectively. Keeping in mind that the latent features are positive, we initialize $\mathbf{x}$ and $\mathbf{y}$ as follows:

$$\widehat{\mathbf{x}} = |UD^{1/2}| \text{ and } \widehat{\mathbf{y}} = |VD^{1/2}|.$$

But keep in mind that CAVI (Algorithm 1) uses the parameters of the Gamma distribution of the latent positions, not the latent positions themselves. Therefore, what we actually need to initialize are the shape and rate parameters of the Gamma distribution. We set the shape parameter to the SVD value and the rate parameter to 1. This setup ensures that the expected value of the Gamma distribution matches the sampled value. Additionally, larger shape parameters increase the variance of the distribution. Intuitively, this allows the algorithm to explore a wider range of potential latent positions when the shape parameter is large, that is, when the expected value of the latent feature is far from 0.

Step 3: Initialize the shape and rate of the spread hyperparameters $\boldsymbol{\zeta}^{(x)}$ and $\boldsymbol{\zeta}^{(y)}$.

By looking at the update equations, we can see that we already have all the quantities that we need in order to compute them. Thus, in this case, we can just use the update equations for the initialization.

Step 4: Initialize the shape and rate of the correction factors $\boldsymbol{\rho}^{(x)}$ and $\boldsymbol{\rho}^{(y)}$.

By examining the update equations, we can see that we have most of the required quantities for their computation. The only missing elements are $\mathrm{ETG}[\rho_{tj}^{(y)}]$ and $\mathrm{ETG}[\rho_{ti}^{(x)}]$. However, once we initialize $\boldsymbol{\rho}^{(x)}$, we have everything needed to compute the update for $\boldsymbol{\rho}^{(y)}$.
Therefore, we only need to estimate $\mathrm{ETG}[\rho_{tj}^{(y)}]$ to use the updates for initialization. The correction factors affect the overall intensity of interactions at various times. An intuitive estimator would be the average interaction of node j at time t, which is $\frac{1}{N_1}\sum_{i=1}^{N_1} A_{tij}$.

Thus, after estimating $\mathrm{ETG}[\rho_{tj}^{(y)}]$, we can initialize $\boldsymbol{\rho}^{(x)}$ and $\boldsymbol{\rho}^{(y)}$ using the update equations.

3.7.2 Convergence Criteria

To check if the algorithm has converged, we will use the ELBO. Since computing the ELBO at each iteration is computationally expensive, we will only compute it every few iterations (e.g. every 5 or 10 iterations).

The **relative absolute change** in the ELBO (**RAC-ELBO**) is defined as:

$$\text{RAC-ELBO}_k \overset{\text{def.}}{=} \frac{|\text{ELBO}_k - \text{ELBO}_{k-1}|}{|\text{ELBO}_{k-1}|}, \text{ where ELBO}_k \text{ is the } k\text{-th computation of the ELBO.}$$

We will terminate the algorithm when $\text{RAC-ELBO}_k < \epsilon$, where ϵ is a small positive number, such as 10^{-5} or 10^{-6}. Additionally, we will set a maximum number of iterations, such as 10^4 or 10^5, to prevent the algorithm from running indefinitely.

3.8 Link Prediction

Given the values of A_{tij} at times $t = 1, \dots, T$, we aim to predict the probability (score) of a link existing between nodes i and j at time $t = T+1$, denoted as $p(A_{t+1,ij} = 1)$, to forecast the future evolution of the network. We can compute this probability using the estimated values of the parameters $\mathcal{P}$.

In our model, we have that:

$$A_{t+1,ij} \mid \rho^{(x)}_{t+1,i}, \rho^{(y)}_{t+1,j}, \mathbf{x}_i, \mathbf{y}_j \sim \text{Bernoulli}\left[1 - \exp(-\rho^{(x)}_{t+1,i}\rho^{(y)}_{t+1,j}\mathbf{x}_i^\top \mathbf{y}_j)\right],$$
$$p\left(A_{t+1,ij} = 1 \mid \rho^{(x)}_{t+1,i}, \rho^{(y)}_{t+1,j}, \mathbf{x}_i, \mathbf{y}_j\right) = 1 - \exp(-\rho^{(x)}_{t+1,i}\rho^{(y)}_{t+1,j}\mathbf{x}_i^\top \mathbf{y}_j).$$

This gives us the probability:

$$p(A_{t+1,ij} = 1) \overset{\text{LTP}}{=} \mathbb{E}_q\left[p\left(A_{t+1,ij} = 1 \mid \rho^{(x)}_{t+1,i}, \rho^{(y)}_{t+1,j}, \mathbf{x}_i, \mathbf{y}_j\right)\right] = 1 - \mathbb{E}_q\left[\exp(-\rho^{(x)}_{t+1,i}\rho^{(y)}_{t+1,j}\mathbf{x}_i^\top \mathbf{y}_j)\right].$$

Of course, this expectation can be approximated by using a Monte Carlo method. Although this would give us an unbiased estimator, it is computationally expensive since we would need to sample from the variational distribution q^* many times. Hence, we propose the following natural estimator:

$$\widehat{p}(A_{t+1,ij} = 1) = 1 - \exp\left\{-\mathbb{E}_q\left[\rho^{(x)}_{t+1,i}\right]\mathbb{E}_q\left[\rho^{(y)}_{t+1,j}\right]\mathbb{E}_q\left[\mathbf{x}_i\right]^\top \mathbb{E}_q\left[\mathbf{y}_j\right]\right\}.$$

Although this estimator is biased since $p(A_{t+1,ij} = 1) \leq \widehat{p}(A_{t+1,ij} = 1)$ (due to Jensen's inequality), it is computationally efficient and works well in experiments [Sanna Passino et al., 2022].

We are able to compute $\mathbb{E}_q[\mathbf{x}_i]$ and $\mathbb{E}_q[\mathbf{y}_j]$, but $\mathbb{E}_q[\rho^{(x)}_{t+1,i}]$ and $\mathbb{E}_q[\rho^{(y)}_{t+1,j}]$ need to be estimated. We may do so by fitting an ARIMA model. However, to ensure these estimates fall within the interval $(0, 1)$, we will apply a reversible transformation σ^{-1} that shifts the data to $\mathbb{R}$. Specifically, we map the data to $\mathbb{R}$ using σ^{-1}, fit an ARIMA model, and then map the predictions back to $(0, 1)$ using σ.

Algorithm 4 outlines the procedure to estimate the probability of an interaction between nodes i and j at time $t = T + 1$.

Algorithm 4: Link Prediction Algorithm in our Model

Input: Estimated variational family q^*, index i, index j.
Output: Estimate of the probability of a link between nodes i and j at time $T+1$.
Step 1: Estimate the latent features to their expected values:

$$\widehat{x}_{ir} = \mathbb{E}_q[x_{ir}] = \lambda^{(x)}_{ir} \;/\; \mu^{(x)}_{ir} \quad \text{and} \quad \widehat{y}_{jr} = \mathbb{E}_q[y_{jr}] = \lambda^{(y)}_{jr} \;/\; \mu^{(y)}_{jr}.$$

Step 2: Estimate the correction factors $\widehat{\rho}^{(x)}_{1,i}, \dots, \widehat{\rho}^{(x)}_{T,i}$ and $\widehat{\rho}^{(y)}_{1,i}, \dots, \widehat{\rho}^{(y)}_{T,i}$ to their expected values:

$$\widehat{\rho}^{(x)}_{ti} = \mathbb{E}_q[\rho^{(x)}_{ti}] = \text{ETG}\left[\rho^{(x)}_{ti}\right] \quad \text{and} \quad \widehat{\rho}^{(y)}_{ti} = \mathbb{E}_q[\rho^{(y)}_{tj}] = \text{ETG}\left[\rho^{(y)}_{tj}\right].$$

Step 3: Map the correction factors to $\mathbb{R}$ by applying σ^{-1}. By fitting an ARIMA model, we obtain predictions for $\sigma^{-1}(\widehat{\rho}^{(x)}_{T+1,i})$ and $\sigma^{-1}(\widehat{\rho}^{(y)}_{T+1,i})$. By reversing the transformation, we obtain estimates for $\widehat{\rho}^{(x)}_{T+1,i}$ and $\widehat{\rho}^{(y)}_{T+1,i}$.
Step 4: Compute the estimate of the probability of a link between nodes i and j at time $T+1$:

$$\widehat{p}(A_{t+1,ij} = 1) = 1 - \exp\left\{-\widehat{\rho}^{(x)}_{T+1,i}\widehat{\rho}^{(y)}_{T+1,i}\widehat{\mathbf{x}}_i^\top \widehat{\mathbf{y}}_j\right\}.$$

return The estimated probability $\widehat{p}(A_{t+1,ij} = 1)$.

A natural choice for σ^{-1} would be the logit function, which is the inverse of the logistic sigmoid function:

$$\sigma\colon \mathbb{R} \to (0,1) \text{ is the } \textbf{logistic sigmoid} \text{ function: } \sigma(x) = \frac{1}{1+\exp(-x)},$$
$$\sigma^{-1}\colon (0,1) \to \mathbb{R} \text{ is the } \textbf{logit} \text{ function: } \sigma^{-1}(p) = \log\left(\frac{p}{1-p}\right).$$

3.9 Evaluation

We are now able to estimate the probability of a link between any two pair of nodes at time $T+1$. In order to evaluate the performance of our model, we can use the ROC curve and AUC. A review of this topic can be found in Appendix B.

Moreover, we can also plot the ELBO and log-likelihood to see how they evolve over time. This will give us an idea of how well the model is fitting the data.

But keep in mind that the formula for the log-likelihood is given by:

$$\log L\left(\Phi \mid \mathbf{A}\right) = \sum_{t,i,j:A_{tij}=1} \log\left\{\exp\left(\rho_{ti}^{(x)}\rho_{tj}^{(y)}\mathbf{x}_i^\top\mathbf{y}_j\right) - 1\right\} - \sum_{t=1}^{T}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{r=1}^{d}\rho_{ti}^{(x)}\rho_{tj}^{(y)}x_{ir}y_{jr}$$

Our model does not infer the correction factors and latent positions directly, but rather the parameters of their Gamma distributions. Hence, we cannot compute the log-likelihood itself, but we may try to compute the expected log-likelihood under the variational approximation:

$$\begin{aligned}\mathbb{E}_q\left[\log L\left(\Phi \mid \mathbf{A}\right)\right] &= \sum_{t,i,j:A_{tij}=1} \mathbb{E}_q\left[\log\left\{\exp\left(\rho_{ti}^{(x)}\rho_{tj}^{(y)}\mathbf{x}_i^\top\mathbf{y}_j\right) - 1\right\}\right] - \sum_{t=1}^{T}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{r=1}^{d}\mathbb{E}_q\left[\rho_{ti}^{(x)}\rho_{tj}^{(y)}x_{ir}y_{jr}\right] \\ &= \sum_{t,i,j:A_{tij}=1} \mathbb{E}_q\left[\log\left\{\exp\left(\rho_{ti}^{(x)}\rho_{tj}^{(y)}\mathbf{x}_i^\top\mathbf{y}_j\right) - 1\right\}\right] - \sum_{t=1}^{T}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{r=1}^{d}\widehat{\rho}_{ti}^{(x)}\widehat{\rho}_{ti}^{(y)}\widehat{x}_{ir}\widehat{y}_{jr}\end{aligned}$$

We do not have an expression for the first term. Although we could compute it using a Monte Carlo method, that would be computationally expensive. Instead, we can estimate it by plugging the estimates for $\rho_{ti}^{(x)}$, $\rho_{tj}^{(y)}$, x_{ir}, and y_{jr} into the formula:

$$\widehat{\mathbb{E}}_q\left[\log\left\{\exp\left(\rho_{ti}^{(x)}\rho_{tj}^{(y)}\mathbf{x}_i^\top\mathbf{y}_j\right) - 1\right\}\right] = \log\left\{\exp\left(\widehat{\rho}_{ti}^{(x)}\widehat{\rho}_{ti}^{(y)}\widehat{\mathbf{x}}_i\widehat{\mathbf{y}}_j\right) - 1\right\}.$$

This is a biased estimator that give us larger values than the true expectation due to Jensen's inequality, since the function $(0,\infty) \to \mathbb{R}, x \mapsto \log\left(\exp(x) - 1\right)$ is concave. However, such an estimator is computationally efficient and works well in experiments [Sanna Passino et al., 2022].

As we will see later, overestimating the expected log-likelihood may cause some oscillatory behaviour when plotting it. This is why we should not rely solely on the log-likelihood to evaluate the model, but rather use it as a reference. Nonetheless, it is still a good indicator of how well the model is fitting the data.

Chapter 4

Simulation Study

Before applying the proposed model on real-world data, we conduct a simulation study in order to evaluate some of its initial settings and discuss potential issues that may arise.

4.1 Simulation of Networks

A **stochastic block model (SBM)**, as introduced in Holland et al. [1983], is a type of statistical model used for community detection in networks. It is particularly useful for understanding the structure of networks by dividing the nodes into groups, or communities, such that the probability of connections between nodes is dependent on the group to which they belong.

Mathematically, there are k_1 clusters of source nodes and k_2 clusters of destination nodes, where k_1 and k_2 are integers greater than or equal to 1. Consequently, all possible edges between nodes can be allocated into a total of $k_1 \times k_2$ different groups.

Algorithm 5 proposes a stochastic block model to simulate the realization $\{\mathbf{A}_t\}$ of an interaction network, where $\mathbf{A}_t \in \{0,1\}^{N_1 \times N_2}$ is the adjacency matrix at time t.

The matrix $B = \{B_{ij}\} \in \mathbb{R}^{k_1 \times k_2}$ represents the interaction matrix between the groups. Here, each element B_{ij} represents an interaction term between the i-th source cluster and the j-th destination cluster, that modifies the probability of an edge existing between a node in the i-th source cluster and a node in the j-th destination cluster. Since B is sampled from a purely continuous distribution, it has full rank with probability 1. By applying truncated SVD with rank $= \min(k_1, k_2)$, we get $B = UDV^T$, and then obtain the 'true' latent features $\mathbf{x} = UD^{1/2}$ and $\mathbf{y} = VD^{1/2}$ imposed by B.

Furthermore, the classifiers $F^{(x)}$ and $F^{(y)}$ determine the cluster membership of each node.

Algorithm 5: Simulation of a network using a stochastic block model.

```
1. Sample the interaction matrix: $B_{ij} \sim \text{Gamma}(\alpha, \beta)$ for $1 \le i \le k_1$ and $1 \le j \le k_2$.
2. Sample the source nodes cluster labels $F_i^{(x)} \sim \text{Discrete}[1, k_1]$ for $1 \le i \le N_1$.
3. Sample the destination nodes cluster labels $F_j^{(y)} \sim \text{Discrete}[1, k_2]$ for $1 \le j \le N_2$.
4. Sample the correction factors $\rho_{ti}^{(x)} \sim \text{Gamma}_{(0,1)}(q_{ti}^{(x)}, r_{ti}^{(x)})$ and $\rho_{tj}^{(y)} \sim \text{Gamma}_{(0,1)}(q_{ti}^{(y)}, r_{ti}^{(y)})$.
5. Simulate the adjacency matrices $\{\mathbf{A}_t\}$:
       $$A_{tij} \sim \text{Bernoulli}\left(1 - \exp\left(-\rho_{ti}^{(x)} \rho_{tj}^{(y)} B_{F_i^{(x)} F_j^{(y)}}\right)\right).$$
```

Here, Discrete$[a, b]$, where a, b are integers, denotes a discrete uniform distribution over the integers in the range $[a, b]$. That means its support is $\{a, a+1, \dots, b\}$, and the probability of each integer is $\frac{1}{b-a+1}$. Furthermore, $\alpha, \beta, \{q_{ti}^{(x)}\}, \{r_{ti}^{(x)}\}, \{q_{ti}^{(y)}\}, \{r_{ti}^{(y)}\}$ are positive real deterministic constants.

In our simulation, we set $T = 20$ time steps, $N_1 = 50$ source nodes, and $N_2 = 40$ destination nodes. For visualization purposes, we are going to work with two clusters, that is, we set $k_1 = k_2 = 2$. Moreover, we are going to set $\alpha = 2$, $\beta = 6$, $q_{ti}^{(x)} = r_{ti}^{(x)} = q_{ti}^{(y)} = r_{ti}^{(y)} = 1$.

4.2 Training the model

After simulating the dataset, we are going to train the model. For the hyperparameters, we are going to use the following values: $a^{(x)} = a^{(y)} = 1$, $b^{(x)} = b^{(y)} = 1$, $c^{(x)} = c^{(y)} = 0.1$, $\alpha^{(x)} = \alpha^{(y)} = 1$, $\beta^{(x)} = \beta^{(y)} = 1$. Since we are working with two clusters, we set the number of latent features to $d = 2$ in order to assess whether the model is able to recover the true clusters. We are going to evaluate the ELBO and log-likelihood at every 2 iterations, set the maximum number of iterations to 10^4, and set the ELBO tolerance to 10^{-6}.

Remember that the proposed model estimates the latent features to their expected values:

$$\widehat{x}_{ir} = \mathbb{E}_q[x_{ir}] = \lambda_{ir}^{(x)} \;/\; \mu_{ir}^{(x)} \quad \text{and} \quad \widehat{y}_{jr} = \mathbb{E}_q[y_{jr}] = \lambda_{jr}^{(y)} \;/\; \mu_{jr}^{(y)}.$$

Figure 4.1 displays the latent representations of the source and destination nodes. We can see that nodes from the same cluster are close to each other, while nodes from different clusters are far apart. This suggests that the model is indeed able to recover the true clusters, and that picking $d = 2$ is a good choice for reducing dimensionality while still keeping enough details about the network structure.

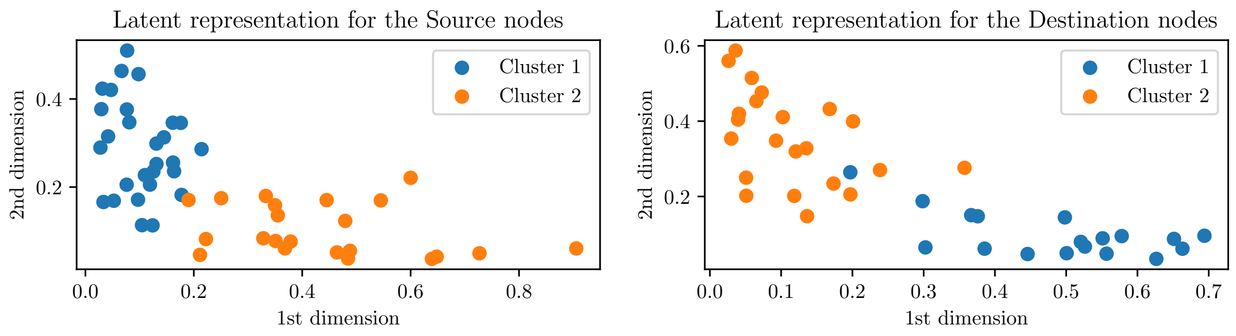


Figure 4.1: Latent representations of source and destination nodes.

We may plot the true and estimated correction factors in order to assess whether the model is able to retrieve them. But we have a problem: $\{\rho_{ti}^{(x)}\}$ and $\{\rho_{tj}^{(y)}\}$ would have the same influence over the network as $\{c \cdot \rho_{ti}^{(x)}\}$ and $\{\rho_{tj}^{(y)}/c\}$, where c is a positive constant.

Therefore, we need to account that there might be a scaling factor present when plotting the true and estimated correction factors. In order to amend this, we propose the following heuristic: we consider that every cluster has its own scaling factor. An intuitive approximation of it would be to consider the ratio of the sum of estimated correction factors and the sum of true correction factors. By dividing the estimated correction factors by this ratio, we bring them on the same scale as the true correction factors.

Let us write this mathematically. We are going to present the heuristic 'normalization' for the source nodes, but the same applies for the destination nodes. Remember that $\widehat{\rho}_{ti}^{(x)}$ is the estimated correction factor, while $\rho_{ti}^{(x)}$ is the true correction factor. We want the scaling factor to have the following property:

$$\sum_{i \in \text{cluster } k} \widehat{\rho}_{ti}^{(x)} = r_k \times \sum_{i \in \text{cluster } k} \rho_{ti}^{(x)}$$

Clearly, the ratio of the two sums satisfies this condition. Therefore, in order to bring the estimated correction factors to the same scale as the true correction factors, we divide them by the ratio of the sums r_k.

We are now able to compare the estimated correction factors and the true correction factors in a meaningful way.

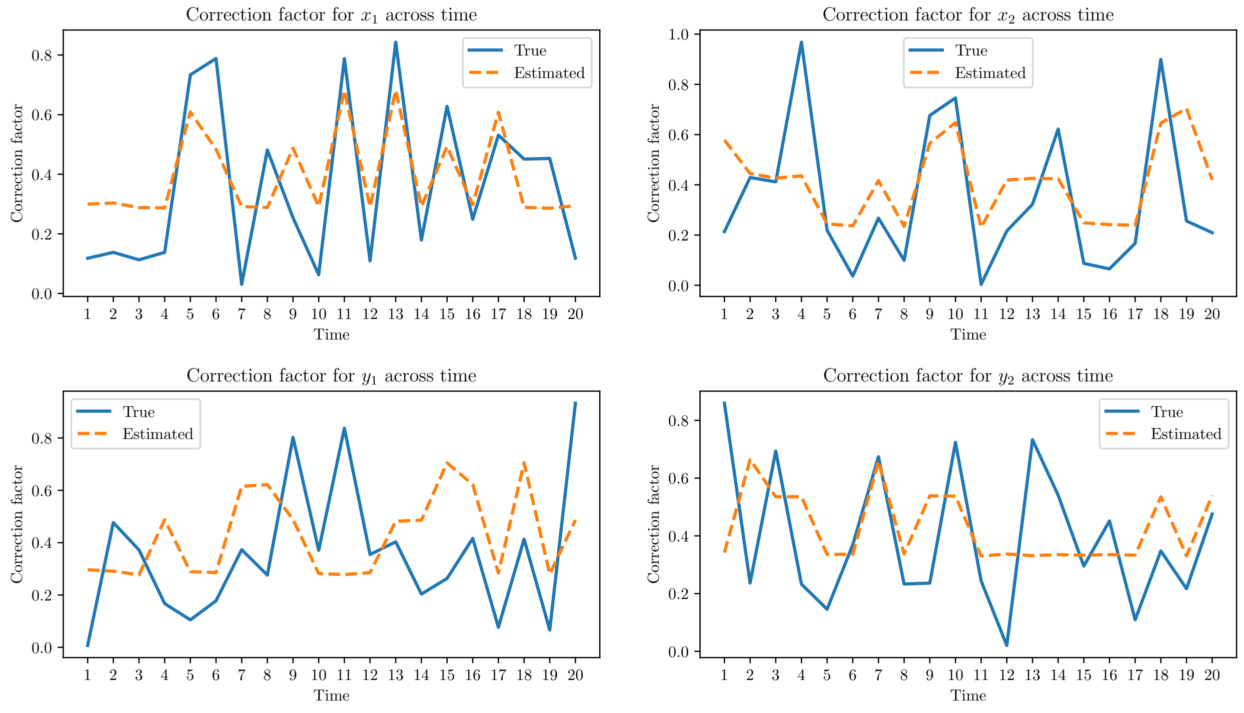


Figure 4.2: Correction factors for the first two source nodes and the first two destination nodes across different times.

Figure 4.2 displays a comparison between the true and estimated correction factors for the first two source nodes and the first two destination nodes. We can see that the model is able to retrieve part of the true correction factors, but it is not able to recover the exact values. This is expected, since the model is not able to recover the true correction factors, but rather the expected values of the latent features, which, in some cases, might not be the best estimate.

4.3 Initialization of Latent Positions

In variational inference, we aim to find a density with a density family $\mathcal{Q}$ that minimizes the KL-divergence between the true posterior and itself (which is equivalent to maximizing the ELBO). Since this is an optimization method, the initialization of parameters can greatly influence the convergence speed and outcome of the algorithm. While we strive to find the global minimum of the KL divergence, the algorithm often gets trapped in local minima, causing the ELBO to stagnate. To explore how different initial values of the latent position affect the algorithm, we will analyse the progress of the ELBO and log-likelihood on our simulated data.

Keeping in mind that, in the proposed model, the latent features need to be positive, we propose the following initializations strategies:

1. **Truncated SVD**: This gives us a latent position encoding that might be close to the true latent positions.

Let $\bar{\mathbf{A}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{A}_t$ be the average interaction matrix. Using truncated SVD with d singular values, we approximate $\bar{\mathbf{A}} \approx UDV^T$, where D is a diagonal matrix with the largest d singular values, and U and V are the corresponding left and right singular vectors, respectively. Keeping in mind that the latent features are positive, we initialize $\mathbf{x}$ and $\mathbf{y}$ as follows:

$$\widehat{\mathbf{x}} = |UD^{1/2}| \text{ and } \widehat{\mathbf{y}} = |VD^{1/2}|.$$

2. **Fixed Constant**: Set $x_{ir} = c_x$ and $y_{jr} = c_y$, where c_x and c_y are positive fixed constants.
3. **Random Uniform**: Sample x_{ir} and y_{jr} from the uniform distribution:

$$x_{ir} \sim \text{Uniform}(u_1^{(x)}, u_2^{(x)}) \text{ and } y_{jr} \sim \text{Uniform}(u_1^{(y)}, u_2^{(y)}),$$

where $u_1^{(x)}, u_2^{(x)}, u_1^{(y)}, u_2^{(y)}$ are positive constants.
4. **Random Beta**: Sample x_{ir} and y_{jr} from the beta distribution:

$$x_{ir} \sim \text{Beta}(b_1^{(x)}, b_2^{(x)}) \text{ and } y_{jr} \sim \text{Beta}(b_1^{(y)}, b_2^{(y)}),$$

where $b_1^{(x)}, b_2^{(x)}, b_1^{(y)}, b_2^{(y)}$ are positive constants.
5. **Random Gamma**: Sample x_{ir} and y_{jr} from the gamma distribution:

$$x_{ir} \sim \text{Gamma}(g_1^{(x)}, g_2^{(x)}) \text{ and } y_{jr} \sim \text{Gamma}(g_1^{(y)}, g_2^{(y)}),$$

where $g_1^{(x)}, g_2^{(x)}, g_1^{(y)}, g_2^{(y)}$ are positive constants.

Recall that the CAVI (Algorithm 1) uses the parameters of the Gamma distribution, not the latent positions directly. Thus, initializing x_{ir} and y_{jr} involves setting the Gamma distribution: the shape parameter is set to the SVD/constant/sampled value, and the rate parameter is set to 1. This way, the expected value of the Gamma distribution matches the sampled value, and larger shape parameters result in greater variance in the distribution. Intuitively, this allows the algorithm to explore a wider range of possible values for the latent positions if the shape parameter is large, that is, when the expected value of the latent feature is far from 0.

For our simulation study, we are going to use the following parameters for initializations: $c_x = c_y = 1$, $u_1^{(x)} = u_1^{(y)} = 0$, $u_2^{(x)} = u_2^{(y)} = 1$, $b_1^{(x)} = b_1^{(y)} = 1$, $b_2^{(x)} = b_2^{(y)} = 1$, $g_1^{(x)} = g_1^{(y)} = 1$, and $g_2^{(x)} = g_2^{(y)} = 1$.

Figure 4.3 display the ELBO and log-likelihood for various initializations during the initial iterations, while Figure 4.4 shows the same metrics throughout the entire training process.

By looking at the plots, we can see that the algorithm converge for all initializations in a reasonable number of iterations, but they do not reach the same ELBO and log-likelihood. A closer look reveals that, in all cases, the algorithm encounters a local maximum, and not all initializations manage to break free from it with the chosen convergence criteria. If we left the algorithm to run for more iterations, the initializations that are stuck might eventually be able to escape the local maximum, but this is not guaranteed. Moreover, notice that the truncated SVD initialization manages to escape the local maximum the quickest.

By looking at the starting points, we can see that the truncated SVD gives us the best one; it has the highest ELBO and a reasonable log-likelihood compared to the others. Intuitively, this might be because the truncated SVD initialization gives us a deterministic latent position encoding that is close to the true latent positions.

Moreover, in large-scale real networks, using random initializations for high-dimensional latent positions can be quite risky. The algorithm might get stuck in a local minimum, or even fail to converge. Therefore, when picking an initialization strategy, we should aim for a deterministic one.

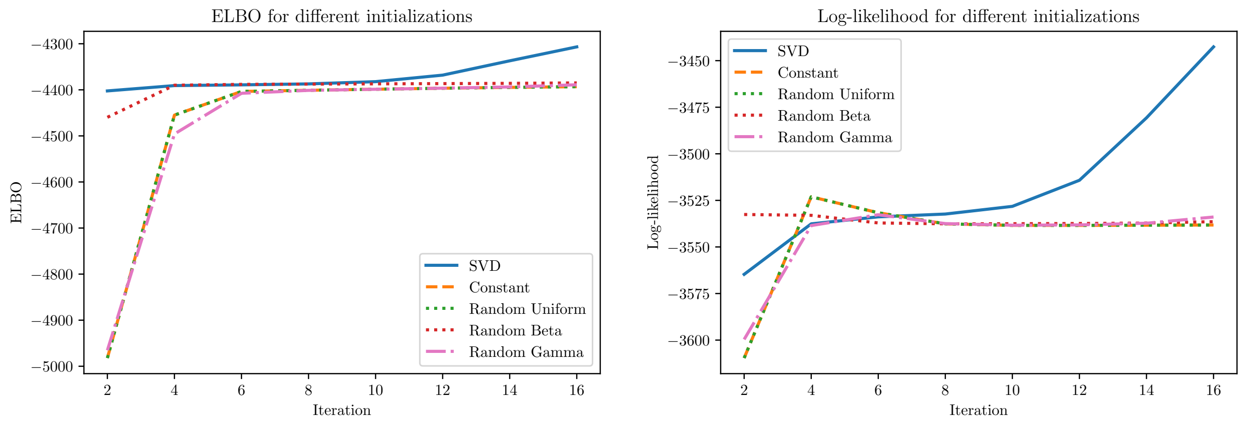


Figure 4.3: Comparison of ELBO and log-likelihood for different initializations during the initial iterations.

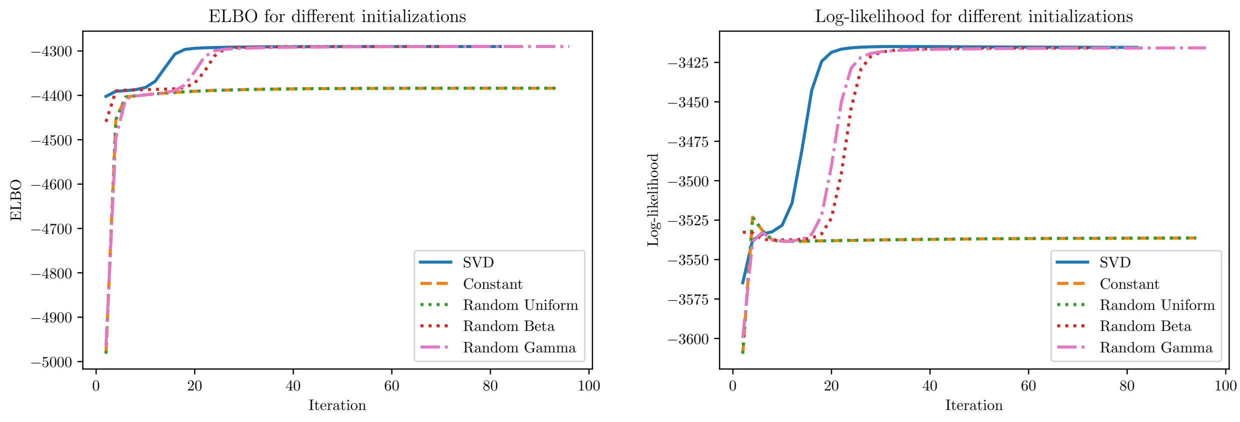


Figure 4.4: Comparison of ELBO and log-likelihood for different initializations throughout the entire training process.

Taking everything into account, we have selected the truncated SVD as our latent position initialization strategy for our algorithm. Its deterministic nature makes it well-suited for scaling to large networks and handling high-dimensional latent positions.

It is important to notice the oscillatory behaviour of the log-likelihood, which is likely caused by the use of a biased estimator for the expected log-likelihood, as discussed in Section 3.9. The ELBO, however, is unaffected by this bias. Therefore, we should rely on the ELBO to check for convergence and use the log-likelihood as a reference.

Chapter 5

Real Data Analysis

In this chapter, we apply our model to predict future connections within real-world network flow data collected from the Huxley Building at Imperial College London. We assess its effectiveness by comparing it with the previously discussed RDPG models, specifically AIP and COSIE. Additionally, we analyse how the model's performance is affected by different days of the week and varying levels of IP activity.

5.1 Exploratory Data Analysis

The data set, referred to as the Huxley NetFlow, includes connections between 164 machines within the Huxley Building at Imperial College and 250 randomly selected internet servers. These connections were recorded as daily snapshots of the temporal graph from 20th January 2020 to 2nd February 2020. For this study, only connections using destination port 443 (HTTPS, secure web browsing) were considered, making this real-world data set a subset of a larger network. If there was no link between two nodes on a given day, the adjacency matrix entry was set to 0. If there was a link, the entry was set 1. For simplicity of notation, we re-index the dates using a numerical time index $t \in \{1, 2, \dots, 14\}$, where $t = 1$ corresponds to 20th January 2020 and $t = 14$ corresponds to 2nd February 2020. Moreover, we denote the adjacency matrices by $\{A_t\}_{t=1}^{14}$.

Date	**Day of the Week**	**Edge Density (%)**
January 20, 2020	Monday	6.28
January 21, 2020	Tuesday	6.65
January 22, 2020	Wednesday	5.70
January 23, 2020	Thursday	6.40
January 24, 2020	Friday	6.89
January 25, 2020	Saturday	3.87
January 26, 2020	Sunday	3.68
January 27, 2020	Monday	7.11
January 28, 2020	Tuesday	6.67
January 29, 2020	Wednesday	5.98
January 30, 2020	Thursday	6.71
January 31, 2020	Friday	6.77
February 1, 2020	Saturday	3.34
February 2, 2020	Sunday	3.57

Table 5.1: Edge density in the Huxley NetFlow data set.

Table 5.1 shows the edge density in the Huxley NetFlow data set for each day, while Figure 5.1 shows the adjacency matrices (as heatmaps) of all the temporal networks.

This dataset is sparse, but not excessively so; typically, computer networks have edge densities smaller than 0.01%. This moderate sparsity is somewhat unusual for network flow data, where most nodes usually interact minimally with each other on a daily basis.

Moreover, the dataset demonstrates a clear weekly periodicity. On weekdays, the edge density is around $5-7\%$, while on weekends, the percentage drops to around $3-4\%$. This is consistent with Huxley Building being a university facility, and data being collected during term time. Staff and students are more likely to use the network during weekdays, when they are on campus for work or study, whereas on weekends, they are more likely to be off-campus, leading to reduced network activity.

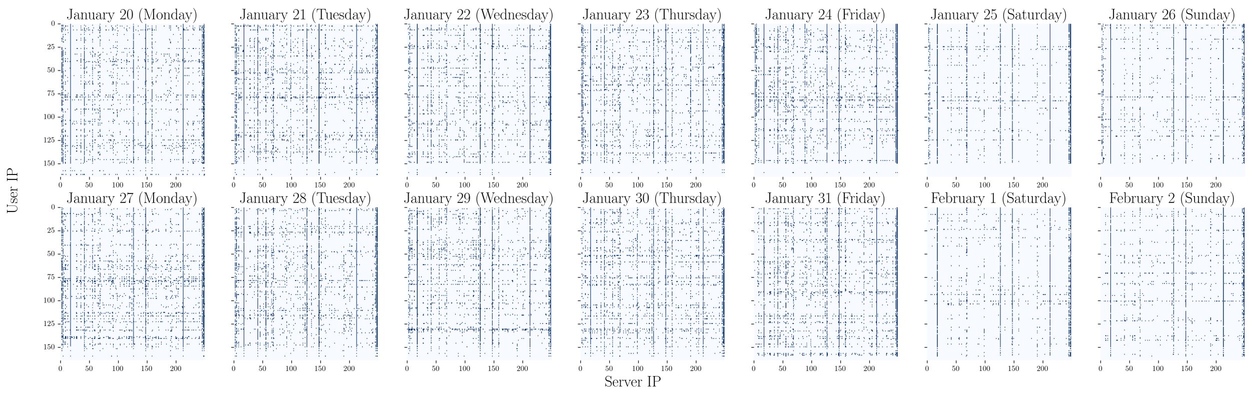


Figure 5.1: Heatmap of the adjacency matrix for the Huxley NetFlow data set. In this visualization, stronger colour represents an active link, while lighter colour represents an inactive link. Clearly, the network is less active on weekends.

In order to train our model, we use the first $T = 10$ temporal networks and test the model on the remaining 4 networks. The training data is used to estimate the model parameters, while the test data is used to evaluate the model's performance.

By following the same methodology as Sanna Passino et al. [2022], Figure 5.2 shows the number of links and the new link rate each day in order to understand the dynamics of the network as it evolves over time. Here, the new link rate is defined as:

$$r_t^{\text{new}} = \frac{\text{Number of active links at time } t \text{ that have never been active before}}{\text{Total number of active links at time } t}$$

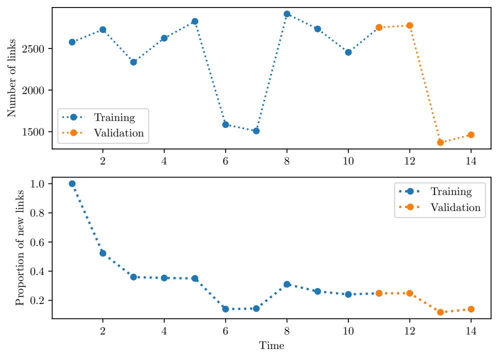


Figure 5.2: Number of links and new link rate in the Huxley NetFlow data set.

As before, we notice a clear weekly trend in the number of links: there are fewer links on weekend compared to weekdays. Moreover, notice that the first new link rate is 1, as all links are new on the first day.

By looking at the new link rate, we see a decreasing tendency over time. This is expected, since the data was collected around the start of the term: staff and students start settling into their routines, and once someone has connected to a server, it is unlikely that they will connect to a different one (unless they have a specific reason to do so, such as a change in their work or study habits). Furthermore, the weekly periodicity is also visible in the new link rate: the rate is lower on weekends compared to weekdays.

We may apply the same ideas to plot the number of active users, active servers, and the new user and server rate each day, which are defined in a similar manner to the new link rate. These plots are shown in Figure 5.3. Here, we consider a user or server to be active if they have at least one active link on a given day.

As before, the weekly periodicity is present for both active users and servers. But we notice something interesting: there are almost no new active users and servers after the first day. That is because most users and servers are active on the first day, so the pool of potential new users and servers is small.

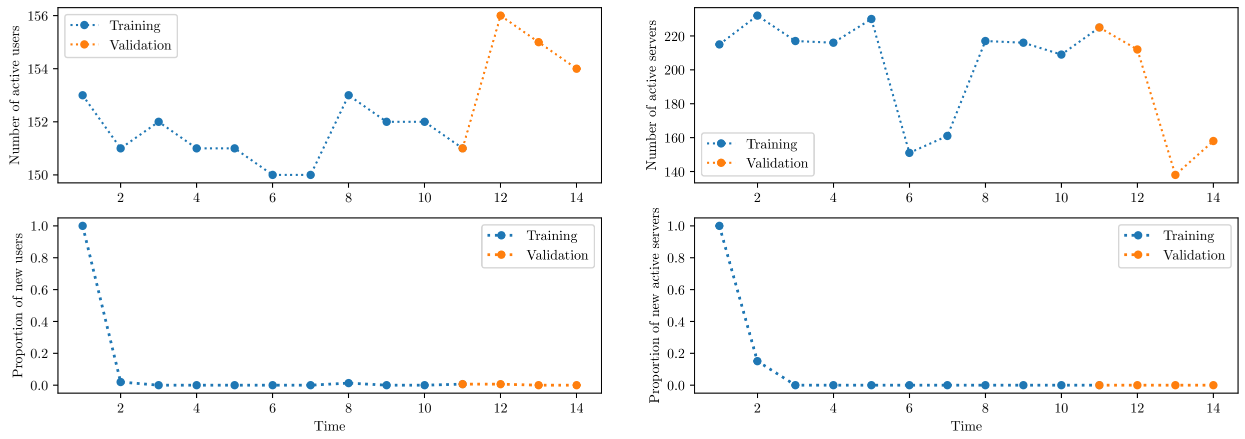


Figure 5.3: Active users and servers in the Huxley NetFlow data set.

5.2 Latent Dimension

When analysing large networks, selecting the appropriate latent dimension for node embedding is essential. This selection helps in identifying lower-dimensional representations of high-dimensional node positions without losing significant information about the network.

In order to do so, we use the **profile likelihood elbow criterion**, as introduced in Zhu and Ghodsi [2006], on all observed adjacency matrices in the training set.

After taking the average interaction matrix across all time points, $\overline{\mathbf{A}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{A}_t$, we apply the Singular Value Decomposition (SVD) method on it. Then, we extract the singular values from this decomposition. These singular values, which represent the amount of variance captured by each dimension, are crucial for determining the optimal latent dimension: larger values correspond to more information about the network structure.

Next, we plot the singular values in descending order. The profile likelihood elbow criterion helps us identify the point, known as the 'elbow', where the rate of decrease in singular values slows down significantly. This elbow point indicates the optimal number of dimensions to retain, balancing the trade-off between capturing sufficient information and minimizing storage requirements.

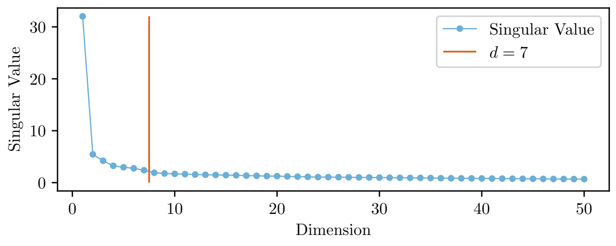


Figure 5.4: Singular Values for the average interaction matrix.

Figure 5.4 shows the top 50 singular values in descending order. By looking at the plot, we can see that a good choice would be $d = 7$ as the elbow point. That is because the rate of decrease in singular values slows down significantly after this point.

Therefore, we choose $d = 7$ as the latent dimension for the node embeddings.

5.3 Training the model

In order to train the model, we are going to use the following hyperparameters: $a^{(x)} = a^{(y)} = 1$, $b^{(x)} = b^{(y)} = 1$, $c^{(x)} = c^{(y)} = 0.1$, $\alpha^{(x)} = \alpha^{(y)} = 1$, $\beta^{(x)} = \beta^{(y)} = 1$. As argued in the previous section, we are going to set the number of latent features to $d = 7$. We are going to evaluate the ELBO and log-likelihood at every 10 iterations, set the maximum number of iterations to 10^4, and set the ELBO tolerance to 10^{-4}. We are going to leave the model to train for the whole period, just to make sure that the RAC-ELBO criterion that we are using is indeed enough to check for convergence.

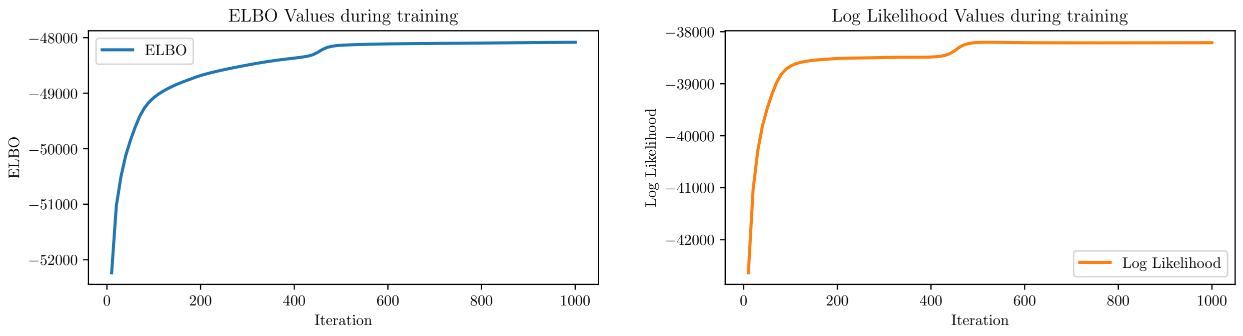


Figure 5.5: Evolution of the ELBO and log-likelihood during training.

Figure 5.5 illustrates the evolution of the ELBO and log-likelihood during training. Both metrics show an increasing trend over time, eventually stabilizing. Although the RAC-ELBO (not shown) confirms that the model has converged, we observe an interesting behaviour in the ELBO plot. Initially, the algorithm seems to converge before iteration 400, but immediately after, there is a noticeable spike. This suggests that the algorithm encountered a local maximum of the ELBO but managed to escape it. Following this spike, no similar fluctuations are observed, and both the ELBO and log-likelihood stabilize, indicating steady convergence.

5.4 Performance Evaluation

To evaluate our model's performance, we will compare it with the RDPG models discussed in Section 2.3.2, by using AUC as the evaluation metric. As baselines, we will use unweighted AIP and COSIE.

The test set includes data from the last four time points: $t = 11$ corresponds to January 30, 2020 (Thursday); $t = 12$ to January 31, 2020 (Friday); $t = 13$ to February 1, 2020 (Saturday); and $t = 14$ to February 2, 2020 (Sunday).

Model	$t = 11$	$t = 12$	$t = 13$	$t = 14$
Proposed Model	**0.888142**	**0.855364**	0.922669	**0.902421**
COSIE	0.886986	0.852873	**0.923021**	0.894219
AIP	0.870586	0.840605	0.91593	0.888248

Table 5.2: AUC scores across different days.

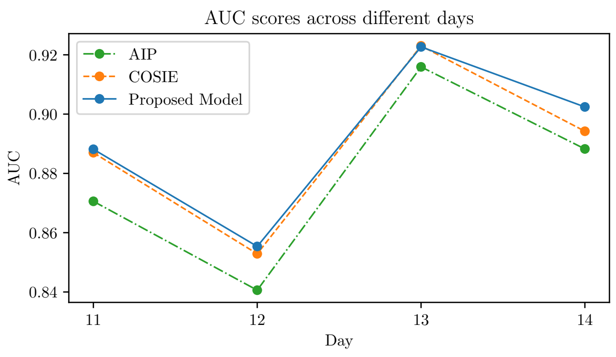


Figure 5.6: AUC scores for the test set across different days.

The AUC scores for the test set are shown in Table 5.2 and Figure 5.6. We observe that AUC scores are lower on weekdays and higher on the weekend. This trend aligns with the network's behaviour, as the number of connections decreases during the weekend, making it easier for the models to predict future connections. This pattern suggests that the models are more effective when there are fewer interactions, possibly due to reduced noise and more stable patterns in the network data during weekends.

Among the three models, AIP consistently performs the worst across all time points, with a noticeable gap between its performance and that of the other two models. This consistent underperformance indicates that AIP is less robust and adaptable to varying network conditions compared to the other models.

Our model outperforms COSIE on three out of the four days, demonstrating its effectiveness and reliability in predicting network connections. Specifically, on days $t = 11$, $t = 12$, and $t = 14$, our model achieves higher AUC scores than COSIE, with a noticeable improvement on day $t = 14$. However, on day $t = 13$, COSIE slightly surpasses our model, albeit by a marginal difference.

In summary, while AIP falls behind in all scenarios, our model and COSIE show strong and competitive performances. The differences between them are relatively small on the first three days, suggesting that they are equally effective during this period. However, on the last day, our model shows a noticeable improvement over COSIE, indicating that our model might be more robust and adaptable to different network conditions.

5.5 Performance Evaluation on Different Activity Levels

Our previous performance evaluation was based on the entire dataset, regardless of the activity levels of the nodes. In this section, we aim to investigate how the performance of our model changes when we split the nodes into groups based on their activity levels. We will then compare the AUC scores of our model with the AIP and COSIE models on these groups. Let us start by defining what we mean by the activity level of a node.

We say that a user i is active at time t if it has at least one connection in the network at that time, that is, if $\sum_{j=1}^{N_2} A_{tij} > 0$. Similarly, we say that a server j is active at time t if it has at least one incoming connection at that time, that is, if $\sum_{i=1}^{N_1} A_{tij} > 0$. We define the activity level of a node as the number of times it is active in the training dataset.

Tables 5.3 and 5.4 show the number of users and servers in each activity level group, respectively, while Figure 5.7 illustrates the distribution of activity levels among nodes in the network. The left chart shows user activity, while the right chart displays server activity.

User Activity Counts	0	1	2	3	4	5	6	7	8	9	**10**
Number of users	6	3	0	3	0	1	0	2	1	4	**144**

Table 5.3: User Activity Counts. The groups in bold are the ones with a significant number of samples.

Server Activity Counts	0	1	2	3	4	5	**6**	**7**	**8**	**9**	**10**
Number of servers	0	0	0	2	3	7	**29**	**42**	**43**	**41**	**83**

Table 5.4: Server Activity Counts. The groups in bold are the ones with a significant number of samples.

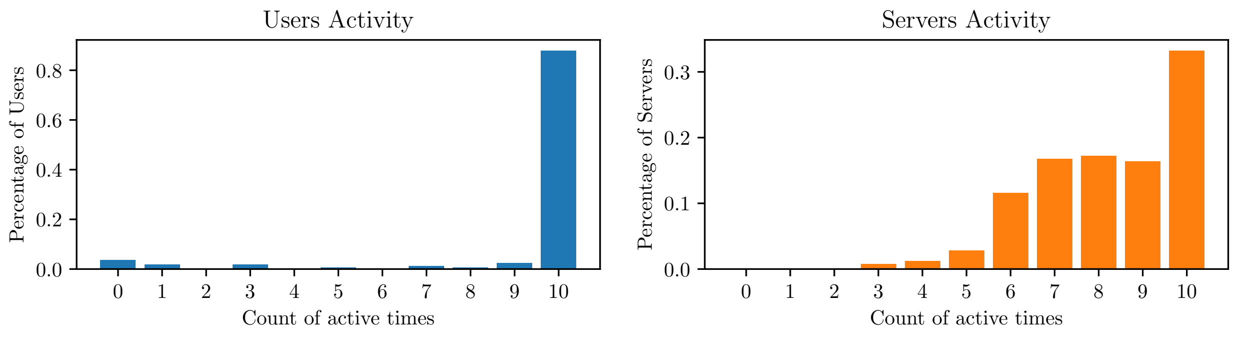


Figure 5.7: Activity levels of the nodes. On the x-axis, we have the number of times a node is active in the training dataset, and on the y-axis, we have the percentage of nodes that fall into each category.

From the user activity chart, it is obvious that the majority of users are active on the network every day, with nearly all users (around 87%) showing activity on all 10 days. This indicates a high level of consistent engagement from users. In contrast, the server activity chart reveals a more varied pattern. Around 33% of servers are active every day, with the majority being active on 6 to 10 days. This suggests that servers do not maintain consistent daily activity, instead exhibiting fluctuations in their activity levels.

This difference in activity patterns between users and servers is expected. Users typically connect to the network daily, leading to higher consistency in their activity. Servers, however, only become active when they receive connections, resulting in a more varied activity profile. Additionally, there are more servers than users, which contributes to the lower percentage of servers being active every day compared to users.

Overall, the plot highlights that users exhibit consistent daily activity, while servers show a broader and more varied range of activity levels. Clearly, the activity levels of users and servers differ significantly, which motivates us to investigate how these differences impact the performance of our model.

However, in order to draw reliable conclusions about an activity level group, we need to have a sufficient number of samples in that group. The only groups that satisfy this criterion are the users with an activity level of 10 and the servers with an activity level in the range of 6 to 10. For all other groups, the number of samples is too small to draw any meaningful conclusions.

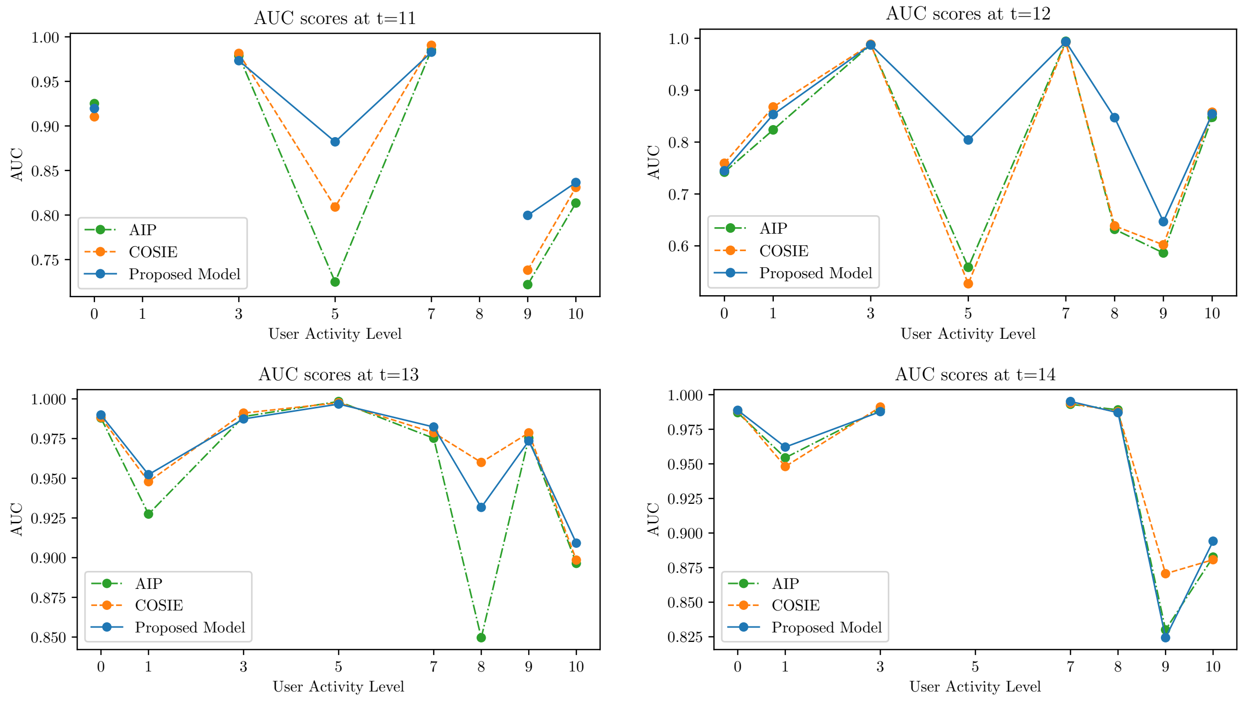


Figure 5.8: AUC scores across different days, as a function of user activity levels.

Figure 5.8 shows the AUC scores across different days, split by user activity levels. As we can see, there are some points missing in the plot and gaps in the lines. This is due to the lack of positive samples in the respective groups, which makes it impossible to calculate the AUC score. This further adds to the previous point that we need a sufficient number of samples in each group to draw reliable conclusions.

Model	$t = 11$	$t = 12$	$t = 13$	$t = 14$
Proposed Model	**0.836538**	0.854246	**0.909148**	**0.894127**
COSIE	0.831032	**0.85751**	0.89848	0.880713
AIP	0.813464	0.847793	0.896396	0.88264

Table 5.5: AUC scores for the user activity level 10 group across different days.

The only group with a sufficient number of samples is the one with user activity level 10. Table 5.5 shows the AUC scores across different days for this group. We get similar results as before: AIP consistently performs the worst, while our model outperforms COSIE on three out of the four days. Even when COSIE surpasses our model, the difference is marginal; we might even say that they perform the same. However, compared to the AUC scores on the whole dataset, there seems to be a higher difference between our model and COSIE on the last two days, that is, on the weekend. This suggests that our model might be more effective at predicting the user's behaviour on the weekend, given that the user has been active every day.

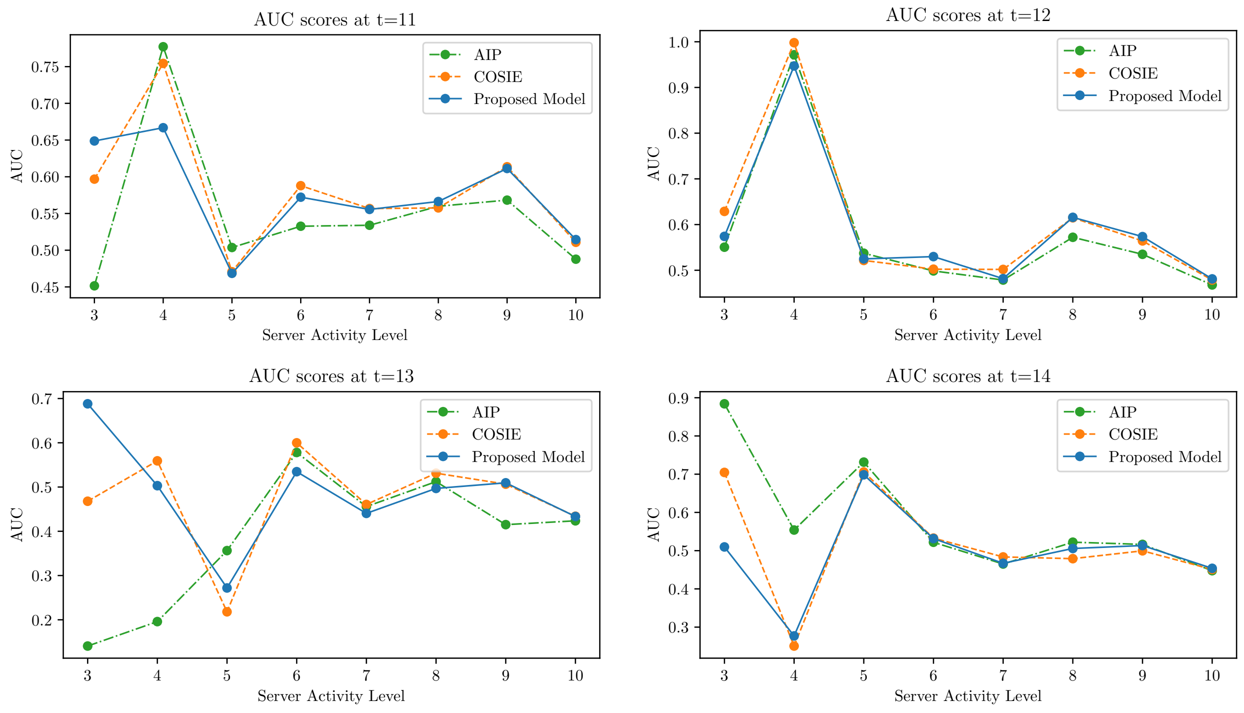


Figure 5.9: AUC scores across different days, as a function of server activity levels.

Day	Model	Activity 6	Activity 7	Activity 8	Activity 9	Activity 10
	Proposed Model	0.572189	0.555529	**0.566312**	0.611252	**0.514491**
$t = 11$	COSIE	**0.587889**	**0.556735**	0.557565	**0.61367**	0.510732
	AIP	0.532652	0.53396	0.559846	0.568212	0.487752
	Proposed Model	**0.529866**	0.481652	**0.615793**	**0.573595**	**0.481089**
$t = 12$	COSIE	0.502363	**0.501822**	0.614830	0.564198	0.478683
	AIP	0.498582	0.478618	0.572149	0.534921	0.467881
	Proposed Model	0.534671	0.44046	0.496871	**0.509518**	0.433381
$t = 13$	COSIE	**0.599905**	**0.460602**	**0.531271**	0.506476	**0.433975**
	AIP	0.578097	0.455798	0.512176	0.415024	0.423646
	Proposed Model	0.531627	0.466975	0.505697	0.513365	**0.453768**
$t = 14$	COSIE	**0.532807**	**0.483607**	0.479059	0.499578	0.450922
	AIP	0.521955	0.465093	**0.522134**	**0.516343**	0.447629

Table 5.6: AUC scores across different days, as a function of server activity levels.

Figure 5.9 shows the AUC scores across different days, split by server activity levels. As we can see, there are no missing points in the plot, which means that, in every case, we have at least one positive sample in the group, which allows us to compute the AUC score. As we said before, the only groups with a sufficient number of samples are the ones with server activity levels in the range of 6 to 10. Table 5.6 shows the AUC scores across different days for these groups.

The results differ significantly from those obtained on the whole dataset. In these cases, all models exhibit poor performance, which might be attributed to the inherent sparsity of the dataset. In a sparse dataset, there is a small number of positive samples, which can lead to significant variability in data distribution across different subsets. This variability can result in different AUC scores, even if the model makes the same predictions for the same samples. Smaller subsets contain fewer non-zero entries, providing limited information and making the model's performance less reliable. Moreover, sparse datasets are more sensitive to noise, and the relevance of features can differ between the whole dataset and specific subsets, further impacting performance.

Notice that we obtain AUC scores below 0.5 for some groups, indicating that the model is performing worse than random guessing. In these instances, flipping the predictions could potentially yield better results. However, even after flipping, the AUC scores remain low, demonstrating poor model performance regardless of the adjustment.

It is important to note that some groups with server activity levels from 0 to 5, which were not considered due to the small number of samples, might still influence the model's performance on the whole dataset. This is because the model is trained on the entire dataset, and the information from these groups is still present in the model. This might be one of the reasons why the AUC scores are lower on these groups compared to the whole dataset.

This situation can also be related to *Simpson's Paradox* [Simpson, 1951]: the marginal effect is not uniformly identical to the joint effect that we see. This means that the trend we observe in the whole dataset changes when we split the dataset into groups. In our case, the model performs well on the whole dataset, but when we split the dataset into groups, the performance of the model decreases. This paradox highlights the critical importance of considering data distribution and context when interpreting statistical results. It illustrates how aggregated data can sometimes reveal trends or patterns that are hidden when examining subsets of the data.

Therefore, in order to draw reliable conclusions, we need to find other ways to evaluate the performance of our model on these groups, as the AUC score might not be the best metric to use in this case.

Chapter 6

Conclusion and future work

In this project, we have started by discussing the importance of link prediction in network analysis, and particularly in cyber-security, where it is crucial to identify nodes that exhibit unusual changes in behaviour. We have discussed the challenges that arise when dealing with time evolving networks, where nodes may vary their activity levels over time, while keeping the same type of connections.

Then, we introduced one of the most popular models for link prediction, the Poisson Matrix Factorization (PMF) model [Gopalan et al., 2015]. By assuming that every node has a latent position in a low-dimensional space, the model is able to capture the similarity between nodes and predict the likelihood of a link between them. However, the PMF model does not account for the temporal dynamics that are present in many real-world networks. In particular, it does not consider that nodes may vary their activity levels over time, while keeping the same type of connections.

To address this issue, we have proposed a dynamic degree-corrected PMF model that extends the original model to account for the temporal dynamics in evolving networks. By introducing time-dependent correction factors for the latent positions, the model is able to capture the changes in node activity levels over time, while keeping the same type of connections. We have formulated the model as a Bayesian hierarchical model, and proposed a variational inference algorithm for fitting the model to large real-world datasets, which is a more scalable alternative to MCMC methods.

After implementing the proposed model in Python, we evaluated its performance on synthetic data, generated from a stochastic block model [Holland et al., 1983]. By plotting the latent positions of the nodes, we demonstrated that the model effectively captures the true clusters of the network. The model also provides reasonable estimates of the correction factors, successfully reflecting changes in node activity levels over time. Moreover, we identified truncated SVD as the optimal method for initializing the latent positions, drawing inspiration from the latent representation offered by the DASE method [Young and Scheinerman, 2007; Athreya et al., 2018].

At the end, we applied the model to a real-world dataset, the Huxley NetFlow dataset, which contains network traffic data collected from the Huxley Building at Imperial College London. We have compared its performance to the RDPG models AIP [Dunlavy et al., 2011] and COSIE [Arroyo et al., 2021], by using the AUC score as an evaluation metric. The model showed promising results, outperforming the AIP in all cases and having similar performance to COSIE. However, by looking at the models' performance on subsets based on IP activity levels, we have noticed a significant drop in performance for all models, which might be caused by the high sparsity of the dataset. This suggests that the AUC score might not be the best metric for evaluating the models in this context, and we have suggested looking for alternatives.

The corresponding code for the model implementation, simulation study, and real-data analysis can be found on GitHub at: `https://github.com/adiboroica/sddc-pmf`.

6.1 Ethical Issues

The dataset we used was collected from the Huxley Building at Imperial College London and contains network traffic data. It consists solely of adjacency matrices, which represent the network at various time points. These matrices provide information about the connections between nodes in the network but do not include any specifics about the users (source nodes) or the servers (destination nodes) involved. This means there is no way to trace the data back to any individual or specific entity.

Given the complete anonymization of the dataset, we believe that using this dataset does not raise any ethical concerns. The focus on purely structural network data without any identifying information about the individuals or entities involved ensures that privacy is maintained and ethical standards are upheld. This approach allows us to conduct our research while respecting the confidentiality and privacy of all parties involved.

6.2 Future Work

The proposed model has shown promising results on synthetic and real-world datasets, but there are still several aspects that could be improved or extended in future work. Here are some suggestions for future research.

Use other evaluation metrics: The AUC score might not be the best metric for evaluating the models in this context, as it does not take into account the sparsity of the dataset. Future work could explore other evaluation metrics, such as precision-recall curves or F1 scores, which might provide a more accurate assessment of the models' performance.

Implement the model on GPU: In order to improve the scalability of the model and make it practical for large real-world datasets, we could implement the variational inference algorithm on GPU. This would allow us to take advantage of the parallel processing power of GPUs and speed up the computation.

Explore alternative methods for modelling the correction factors: The link prediction algorithm that we have proposed maps the correction factors to the real line (by using the logit function), fits an ARIMA model to the mapped values, and then maps the predicted values back to the unit interval (by using the logistic function). This approach might not be the most efficient way to fit the correction factors in all cases.

For instance, we might consider other functions that map the unit interval to the real line and are invertible. Alternatively, we could adjust the hyperparameters of the ARIMA model or fit a different time series model altogether. Another approach could be to discard the idea of mapping correction factors to the real line and instead fit a neural network directly to the correction factors, ensuring that its output remains within the unit interval.

Change the dependency modelling between correction factors: In the current approach, we fit an ARIMA model to the transformed correction factors. Hence, our predictions for the future correction factors are based on the past values of the correction factors. We could try to incorporate this dependency directly into the model's structure. For example, by using a Bayesian hierarchical model representation, here is how such a model could look like:

$$\begin{aligned}
A_{i,j,t} \mid \rho_{i,j}, \rho_{j,t}, \boldsymbol{\mu}_i, \boldsymbol{\mu}_j &\sim \text{Bernoulli}(1-\exp\{-\rho_{i,t}\rho_{j,t}\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j\}), \\
\mu_{i,k} &\sim \text{Gamma}(a, \zeta_i), \quad k = 1, \dots, d, \\
\zeta_i &\sim \text{Gamma}(b, c), \\
\rho_{i,t} &= \sigma\{\sigma^{-1}(\rho_{i,t-1}) + \epsilon_{i,t})\}, \quad \epsilon_{i,t} \sim \mathcal{N}(0, \tau^2),
\end{aligned}$$

where a, b, c, τ are constants, and $\sigma \colon \mathbb{R} \to (0,1), \sigma(x) = \frac{1}{1+e^{-x}}$ is the logistic sigmoid function.

Here, $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ are the latent positions of the source and destination nodes, respectively, and $\rho_{i,t}$ and $\rho_{j,t}$ are the correction factors for the source and destination nodes at time t.

The latent positions are modelled as Gamma distributed random variables, to ensure that they are positive. Each node has its own rate parameter, ζ_i, to account for different levels of variability in the latent positions. To ensure that the rate parameter is positive, we model it as a Gamma distributed random variable with shape and rate parameters b and c, respectively.

We want the correction factor $\rho_{i,t}$ at time t to depend on its past value, thereof, one approach would be to start with the correction factor $\rho_{i,t-1}$ at time $t-1$, map it to the real line using σ^{-1}, add a normally distributed noise term $\epsilon_{i,t}$, and then map the result back to the unit interval using σ. This process keeps the correction factors within the unit interval while building the dependency between them directly into the model's structure.

To fit this model, we could explore scalable inferential techniques, such as the approximate variational inference method introduced in Charlin et al. [2015].

Bibliography

Arroyo, J., Athreya, A., Cape, J., Chen, G., Priebe, C. E., and Vogelstein, J. T. (2021). Inference for multiple heterogeneous networks with a common invariant subspace. *Journal of Machine Learning Research*, 22(142):1–49.

Athreya, A., Fishkind, D. E., Levin, K., Lyzinski, V., Park, Y., Qin, Y., Sussman, D. L., Tang, M., Vogelstein, J. T., and Priebe, C. E. (2018). Statistical inference on random dot product graphs: a survey. *Journal of Machine Learning Research*, 18(226):1–92.

Benchettara, N., Kanawati, R., and Rouveirol, C. (2010). Supervised machine learning applied to link prediction in bipartite social networks. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 326–330.

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.

Caron, F. and Fox, E. B. (2017). Sparse Graphs Using Exchangeable Random Measures. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(5):1295–1366.

Casella, G. and Berger, R. L. (2002). *Statistical Inference*. Duxbury, 2nd edition.

Charlin, L., Ranganath, R., McInerney, J., and Blei, D. M. (2015). Dynamic poisson factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, page 155–162, New York, NY, USA. Association for Computing Machinery.

Chen, B., Li, F., Chen, S., Hu, R., and Chen, L. (2017). Link prediction based on non-negative matrix factorization. *PLOS ONE*, 12(8):1–18.

Clauset, A., Moore, C., and Newman, M. E. J. (2008). Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101.

Dunlavy, D. M., Kolda, T. G., and Acar, E. (2011). Temporal link prediction using matrix and tensor factorizations. *ACM Trans. Knowl. Discov. Data*, 5(2).

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874. ROC Analysis in Pattern Recognition.

Gao, S., Denoyer, L., and Gallinari, P. (2011). Temporal link prediction by integrating content and structure information. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, page 1169–1174, New York, NY, USA. Association for Computing Machinery.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741.

Gong, C. and Huang, W.-b. (2017). Deep dynamic poisson factorization model. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1665–1673, Red Hook, NY, USA. Curran Associates Inc.

Gopalan, P., Hofman, J. M., and Blei, D. M. (2015). Scalable recommendation with hierarchical poisson factorization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, UAI'15, page 326–335, Arlington, Virginia, USA. AUAI Press.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Hoff, P. D., Raftery, A. E., and Handcock, M. S. (2002). Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098.

Holland, P. W., Laskey, K. B., and Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137.

Hosseini, S. A., Alizadeh, K., Khodadadi, A., Arabzadeh, A., Farajtabar, M., Zha, H., and Rabiee, H. R. (2017). Recurrent poisson factorization for temporal recommendation. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 847–855, New York, NY, USA. Association for Computing Machinery.

Karrer, B. and Newman, M. E. J. (2011). Stochastic blockmodels and community structure in networks. *Phys. Rev. E*, 83:016107.

Kim, B., Lee, K. H., Xue, L., and Niu, X. (2018). A review of dynamic network models with latent variables. *Statistics Surveys*, 12(none):105 – 135.

Li, L. (2023). Variational inference in dynamic poisson matrix factorisation for link prediction. Imperial College London M4R Thesis.

Newman, M. (2018). *Networks*. Oxford University Press.

Roberts, G. O. and Rosenthal, J. S. (2004). General state space markov chains and mcmc algorithms. *Probability Surveys*, 1:20–71.

Sanna Passino, F., Bertiger, A. S., Neil, J. C., and Heard, N. A. (2021). Link prediction in dynamic networks using random dot product graphs. *Data Min. Knowl. Discov.*, 35(5):2168–2199.

Sanna Passino, F., Turcotte, M. J. M., and Heard, N. A. (2022). Graph link prediction in computer networks using Poisson matrix factorisation. *The Annals of Applied Statistics*, 16(3):1313 – 1332.

Scheinerman, E. R. and Tucker, K. (2010). Modeling graphs using dot product representations. *Computational Statistics*, 25(1):1–16.

Sewell, D. K. and Chen, Y. (2015). Latent space models for dynamic networks. *Journal of the American Statistical Association*, 110(512):1646–1657.

Shi, X. (2022). Modelling node dynamics in binary poisson matrix factorisation for graph link prediction. Imperial College London MSc Thesis.

Simpson, E. H. (1951). The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(2):238–241.

Wolfram Research, Inc. (2001). Derivative of the upper incomplete gamma function with respect to the first argument. Available online: `https://functions.wolfram.com/GammaBetaErf/Gamma2/20/01/01/0002/` [Accessed: 2024-06-03].

Young, S. J. and Scheinerman, E. R. (2007). Random dot product graph models for social networks. In Bonato, A. and Chung, F. R. K., editors, *Algorithms and Models for the Web-Graph*, pages 138–149, Berlin, Heidelberg. Springer Berlin Heidelberg.

Zhou, T., Lü, L., and Zhang, Y.-C. (2009). Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630.

Zhu, M. and Ghodsi, A. (2006). Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics & Data Analysis*, 51(2):918–930.

Appendix A

Markov Chain Monte Carlo (MCMC)

Markov Chain Monte Carlo (MCMC) methods [Roberts and Rosenthal, 2004] are a family of algorithms used to generate samples from a probability distribution where direct sampling is difficult, particularly when dealing with high-dimensional spaces where traditional methods become impractical. MCMC allows for the estimation of the distribution of an unknown parameter by constructing a Markov chain that has the desired distribution as its equilibrium distribution. To briefly recapitulate the key points:

- **Markov Chain**: This is a sequence of random variables where each variable is dependent only on the immediate previous variable. This 'memoryless' property simplifies the process of analysing the chain's behaviour over time.

- **Monte Carlo Simulation**: This is a technique that uses repeated random sampling to obtain numerical results. It is often used to estimate complex integrals or the properties of distributions.

MCMC combines these two methods. It constructs a Markov Chain in such a way that, after a number of steps, it will provide samples from the desired distribution. Common algorithms for MCMC include the Metropolis-Hastings (MH) algorithm [Hastings, 1970] and Gibbs Sampling [Geman and Geman, 1984].

Algorithm 6 presents the Metropolis-Hastings (MH) method.

Algorithm 6: Metropolis Hastings

```
/* $p_*(\cdot)$ is the target (unnormalized) distribution          */
/* $q(\cdot \mid \cdot)$ is the proposal                            */
Input: The number of samples $N$, Starting point $X_0$.
for $n = 1$ to $N$ do
    Propose (sample): $X' \sim q(\cdot|X_{n-1})$. Accept the sample $X'$ with probability:
        $$\alpha(X_{n-1}, X') = \min\left\{1, \frac{p_*(X')q(X_{n-1}|X')}{p_*(X_{n-1})q(X'|X_{n-1})}\right\}.$$
    Otherwise, reject the sample and set $X_n = X_{n-1}$.
end
Discard: First burnin samples.
return The remaining samples.
```

Algorithm 7 presents the Gibbs Sampling method, which is a special case of MH in which the proposal is the exact conditional distribution of the variable being sampled.

Algorithm 7: Gibbs Sampling

```
/* $p_*(\cdot)$ is the target (unnormalized) multivariate distribution. */
Input: The number of samples $N$, starting point $X_0 \in \mathbb{R}^k$.
for $n = 1$ to $N$ do
    for $j = 1$ to $k$ do
        Sample $X_{n,j}$ (the $j$-th component at the $n$-th iteration) from the conditional distribution
          $p_*(X_{n,j}|X_{n,1}, \ldots, X_{n,j-1}, X_{n-1,j+1}, \ldots, X_{n-1,k})$.
    end
end
Discard: First burnin samples.
return The remaining samples.
```

A.1 Advantages and Disadvantages

MCMC offers several **advantages**:

- **Handling Complex Distributions**: MCMC excels at sampling from complex, high dimensional probability distributions that are difficult to tackle with other methods. This makes it invaluable in fields like Bayesian statistics.
- **Flexibility**: It can be applied to a wide variety of problems, including those with non-standard and intricate posterior distributions, without needing major modifications.
- **Convergence**: Given enough time, MCMC methods can approximate the desired distribution regardless of the initial values. This is particularly useful when the shape of the distribution is unknown or hard to approximate.
- **Incorporating Prior Information**: It naturally allows the incorporation of prior knowledge into the statistical analysis through Bayesian frameworks.
- **Quantifying Uncertainty**: It provides a straightforward way to quantify uncertainty in parameter estimates and predictions, which is crucial in many scientific and decision-making processes.

While there are numerous advantages of using MCMC methods, they also come with some **drawbacks**:

- **Computational Intensity**: MCMC algorithms can be computationally expensive, especially for complex models or high-dimensional data. This can lead to long run times and high resource usage.
- **Computational Burden**: When closed-form solutions are not readily available, additional steps (like numerical approximations) may be required, adding to the computational complexity.
- **Convergence Issues**: Determining when the Markov chain has converged to the target distribution can be challenging. If convergence is not achieved, the results may be unreliable.
- **Initial Burn-in Period**: A 'burn-in' period is often required, where initial samples are discarded. Choosing the length of this period is not straightforward and can affect the results.
- **Dependence on Initial Values**: The starting point of the chain can affect how quickly convergence occurs, and in some cases, different starting points can lead to different results.
- **Limited Scalability**: MCMC methods may not scale well with extremely large datasets or models with a vast number of parameters.

These disadvantages limit the applicability of MCMC methods in certain scenarios, in which case we would need to use other sampling techniques.

Appendix B

ROC and AUC score

The problem of predicting links is essentially a binary classification problem in which a link's presence is represented by 1, and a link's absence is represented by 0. In order to examine a model's performance, we need to find an appropriate accuracy measurement. In many relevant studies, particularly in the field of cyber-security link prediction, the accuracy of the model is measured by plotting the ROC (Receiver Operating Characteristic) curve [Fawcett, 2006] and computing the AUC (Area Under the ROC curve). The AUC metric effectively measures the likelihood that, given one randomly chosen positive instance (a true case) and one randomly chosen negative instance (a false case), the model will assign a higher probability (or score) to the positive instance than to the negative one.

In order to define the ROC curve and AUC, we need to introduce the **confusion matrix**, which is a tool used for evaluating the performance of a binary classification model. Table B.1 gives its entries. Using the confusion matrix, we can define the TPR and FPR, which are used to define the ROC curve and AUC.

		Predicted Value	
		Positive	Negative
Actual Value	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table B.1: Confusion Matrix

1. **True Positive Rate (TPR)**: Also known as Sensitivity, Recall, or Hit Rate, TPR measures the proportion of actual positives that are correctly identified as such. In other words, it answers the question: 'Out of all the actual positives, how many did the model correctly identify?'. Mathematically, TPR is calculated as:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

   A higher TPR means the model is more effective at catching positives.

2. **False Positive Rate (FPR)**: It measures the proportion of negative cases that are mistakenly classified as positive. In other words, it answers the question: 'Out of all the actual negatives, how many did the model classify as positive?'. Mathematically, FPR is calculated as:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}}.$$

   A lower FPR indicates that the model is better at avoiding false alarms.

Remember that, in a binary classification problem, our model assigns a probability (score) to each instance. In order to make predictions, we need to set a threshold: if the probability is above the threshold, we predict the instance as positive; otherwise, we predict it as negative. For example, if the threshold is 0.5, we predict all instances with a probability greater than 0.5 as positive and the rest as negative.

Why would we use a threshold different that 0.5 when making predictions? In many real-world scenarios, the cost of false positives and false negatives is not the same. For example, in a fraud detection system, missing a fraudulent transaction (false negative) might be much more costly than flagging a legitimate transaction as fraudulent (false positive). Adjusting the threshold allows you to balance these costs based on their relative importance.

The ROC curve and AUC score are useful tools for evaluating the performance of a binary classifier across different thresholds. Let us discuss them in more detail.

1. **ROC (Receiver Operating Characteristic) Curve**: It is a graphical representation of the performance of a binary classifier. By varying the threshold, we can generate different sets of TP, FP, TN, and FN, which gives us multiple pairs of TPR and FPR.

   The ROC curve plots these pairs of TPR and FPR for different thresholds, which helps us visualize the trade-off between the true positive rate and the false positive rate. The ROC curve is a useful tool for selecting the best model, as it allows us to compare different models based on their performance.

2. **AUC (Area Under the ROC Curve)**: This is a measure of the entire two-dimensional area underneath the entire ROC curve. It provides an aggregate measure of performance across all possible classification thresholds. The AUC ranges from 0 to 1.

   - An AUC close to 1 indicates a model with good measure of separability, that is, it is able to distinguish between the classes effectively.
   - An AUC of 0.5 suggests no discriminative power, equivalent to random guessing.
   - An AUC close to 0 suggests that the model is consistently making wrong predictions. Since it is almost always predicting the opposite of the actual class, we can just flip the predicted classes and we would get a model with an AUC close to 1, which is a model with good measure of separability.

See Figure B.1 for a visual representation of the AUC and ROC.

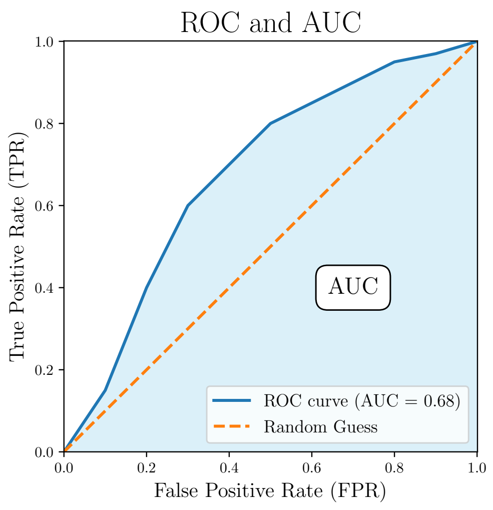


Figure B.1: ROC curve with the AUC shaded. The 'Random Guess' represents a classifier that randomly guesses the class.

How can we estimate the AUC in practice? Due to the vast number of nodes, calculating the TPR and FPR for all possible thresholds is computationally expensive. Instead, we can estimate the TPR and FPR by sampling a subset of the data.

This approach is common in link prediction tasks, especially when dealing with sparse graphs where the number of negative instances significantly exceeds the number of positive instances. By assuming that the dataset is predominantly composed of negative instances, we use all the positive instances and a sample of the negative instances to estimate the AUC. This method allows us to obtain an accurate estimate of the AUC without excessive computational cost. The large number of negative instances ensures that this estimate is reliable.