Joint Inference of Structure and Diffusion in Partially Observed Social Networks

Maryam Ramezani*, Amirmohammad Ziaei†, and Hamid R. Rabiee† Senior Member, IEEE

Abstract—Access to complete data in large scale networks is often infeasible. Therefore, the problem of missing data is a crucial and unavoidable issue in analysis and modeling of real-world social networks. However, most of the research on different aspects of social networks do not consider this limitation. One effective way to solve this problem is to recover the missing data as a pre-processing step. The present paper tries to infer the unobserved data from both diffusion network and network structure by learning a model from the partially observed data. We develop a probabilistic generative model called “DiffStru” to jointly discover the hidden links of network structure and the omitted diffusion activities. The interrelations among links of nodes and cascade processes are utilized in the proposed method via learning coupled low dimensional latent factors. In addition to inferring the unobserved data, the learned latent factors may also help network classification problems such as community detection. Simulation results on synthetic and real-world datasets show the excellent performance of the proposed method in terms of link prediction and discovering the identity and infection time of invisible social behaviors.

Index Terms—Information Diffusion, Partially Observed Social Network, Network Structure.

1 INTRODUCTION

Social networks are essential platforms for the interaction of people by explicit link through following each other as well as implicit connection by sharing information. The widespread use of social networks by increasing the number of users, the large number of interactions between them, and the amount of information propagation over these networks has led to a line of research focused on analyzing and modeling these networks. The target audiences for the results of these research are either (1) the owners of these platforms (e.g., Twitter, Instagram) who own the data, or (2) the third party enterprises with customers that are users of these platforms (e.g., advertising companies, product providers, news analysts). Solving the problem of missing data for the latter community is the focus of this paper. In the context of large scale social media, data collection is a massive, expensive, and time-consuming task for the third party companies. Typically, social datasets are provided for a limited time period and include a subset of users for specific applications. In practice, having access to complete data of a network is impossible even for a short period of time, and we often observe a partial subset of social data, because of the following reasons: (1) API call restrictions: Most social network platforms provide public API for their data access in well-defined formats and automatically trigger a rate-limiting mechanism. (2) Rate-limit for web crawler: Websites usually set a limited number of requests for fetching data per IP address. When a crawler is scraping the site, a temporary ban is done when its request rate goes above the predefined limit. (3) Sampling technique: Due to the large volume and variety of data in a network, sampling methods are used for prioritization. Depending on the sampling method, missing data arise through the collection process. (4) Protecting privacy: Statistics have shown that private accounts and rates of private activities are steadily rising by users of social networks. Since web crawlers do not have access to the information of the private accounts, the collected dataset does not represent the complete data. Although lots of researches have been done on different aspects of social networks, most of them do not consider the problem of missing data. Hence, the completeness assumption in those social network methods affects their performance on real data. To alleviate this problem, some methods utilize a pre-processing algorithm on the input datasets to obtain a set with the least missing data to compensate for their completeness assumption. The main question is: How can we apply the existing methods on the original crawled data of social networks that contain missing data? In this paper, we try to provide an answer to this question for a specific type of data (graph structured data with diffusion information), by proposing a generative probabilistic method called DiffStru.

A network can be modeled by a graph including users and their interactions. The activity of users during time is another data that represents a kind of phenomenon over network named diffusion. Diffusion is usually known as a set of propagation processes called cascade. A cascade is basically the process of transmitting information from one user to another connected user, in a network. In a cascade, a user is said to be infected if it receives a republished information from another user. In this context, infection time is the time of user activity in publishing or republishing information in a cascade. It is good to note that in most of the social platforms, name of nodes and their infection time are the only accessible data from cascades and other additional data such as the infection path and the trace of who infected whom is unknown.

Even if we consider a set of nodes in data collection and

* M. Ramezani, A. Ziaei, and H.R. Rabiee are with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. Email:maryam.ramezani@sharif.edu, ziae@ce.sharif.edu, rabiee@sharif.edu
† Senior Member, IEEE
obtain the links between them for a time interval, we have some unobserved links due to the aforementioned limitations. On the other hand, unobserved information in diffusion data appears as missing activity of nodes in each cascade. Figure (1) illustrates the problem we are trying to solve. Given a limited time period having a partially observed graph structure of the network and diffusion information containing the same set of users, our goal is completing the missing data. We make the following assumptions: (1) We assume that the data is collected in a limited time interval, such that the network structure is static and there is no timestamp for the creation of the link, in that interval. Moreover, with this assumption, the collected diffusion information during the crawling is done over a static structure. Therefore, the network links are also static, and the data is fixed during interactions. On the other hand, a cascade has a limited lifespan and spreads over hours to days, so the sequence from beginning to end of a cascade is usually fixed within a limited data collection interval. (2) We are not interested in hidden nodes that do not have any global or local signals in structure or diffusion data. However, we can somewhat handle users with private accounts. Although we cannot observe their profiles and activities, but we can obtain their interactions from the list of following/followers of their visible neighbors. Therefore, the one-step neighbors of visible users are not hidden while we are not engaged in any hidden neighbors of private users. (3) We consider that the source and mediators of an activity, but we can obtain their interactions from the list of following/followers of their visible neighbors. Therefore, the one-step neighbors of visible users are not hidden while we are not engaged in any hidden neighbors of private users. (3) We consider that the source and mediators of an information dissemination process are internal factors of the network, and any external agents are ignored.

As mentioned before, incomplete data in the collected set from a social network is inevitable, and missing data can significantly affect the difference between the output of methods and what actually happens in the real world [2]. Considering the missing data in two levels of structure and diffusion at the same time, makes the data inference challenging. Here, we try to make the inference tractable by selecting the appropriate distributions on the data. We present a novel generative model for jointly inferring the partial network structure and the information diffusion. The contributions of this paper are as follows:

- Tackling the problem of missing data in real world social datasets by proposing a new probabilistic generative model to jointly discover the hidden links of network structure and omitted diffusion activities.
- Investigating the joint properties of structure and diffusion via probabilistic matrix factorization.
- Inferring the missing links of a network structure and the missing activities of nodes when a partial graph and a set of cascades with missing data are observable.
- Inferring the diffusion behavior of users even when the user page is private, and we have no information about her/his activities.
- Demonstrating that the low-dimensional representations for characteristics of users and cascades during the inference of DiffStru can be widely used in embedding and classification problems.

The rest of the paper is organized as follows: In Section (2), we review the related work, while Section (3) presents the problem statement. We present the propose model in Section (4). In Section (5), we present the empirical results on synthetic and real-world datasets. Finally, we present the conclusion with a discussion of some future work in Section (6).

### 2 RELATED WORK

Partially observed data has received an enormous attention in previous studies of social networks. Some works have focused on measuring the impact of missing data on properties of network and diffusion process. In [3], the properties of full cascade is estimated given an incomplete cascade with complete knowledge of the network structure. The authors in [2] study the characteristics of diffusion process over the partial structure of the network. Non-negative tensor factorization (NTF) tries to model the snapshots of incomplete network and builds a surrogate network similar to the ground-truth network in size and distribution of simulated epidemic. On the other hand, Its extended model named joint NTF, uses coupled complete information [4]. Finally, some works have also explored estimating the statistic properties of the global network from partial data [5]. As shown in Table (1), the related work to our problem includes a wide range of research - Part 1 - Part 2 - Part 3 - Part 4 - Part 5 - Part 6

| Diffusion | Structure | T | P | N |
|-----------|-----------|---|---|---|
| C | Network completion | | | | Structure Inference |
| P | Cascade Prediction | Focus of this paper | Cascade Prediction |
| N | Diffusion Modeling | Link Prediction | Network completion |
from network completion to link and cascade prediction. Nevertheless, none of these works are entirely in line with the proposed method. The focus of this paper is on simultaneous inference from both structure and diffusion data, and currently, there is no previous work that is precisely related to the goals of the proposed method. Here, we summarize the algorithms with the assumption of incomplete data, either in structure or diffusion data.

**Network completion:** This category aims to recover the missing parts of a network. The omitted part can include nodes and links. KronEM predicts them by combining an Expectation-Maximization (EM) with the Kronecker model of graphs. Some works utilize the side information such as node attribute or pairwise similarity between nodes to complete the network.

**Link prediction:** There are numerous proposed models to predict lost links in a network when all nodes and some links are present, including, traditional supervised and unsupervised learning methods, probabilistic stochastic block model, matrix or tensor factorization, locally-based algorithms, and deep learning based approaches. When a partial network is observable on both of the present and absent links, a link prediction matrix factorization with features of nodes in order to predict unobserved links. Uses the diffusion features such as interactive activities of nodes in cascades against topology features for link prediction. Link prediction is modeled with matrix factorization using the similarity of rewetting information between the pairs of users.

**Structure inference:** When the interaction topology of a network is entirely unreachable, an inference approach is put forward using visible independent measurements, utilizing the process over the network or with the help of received signals from nodes to infer their connections. Many attempts have been made to solve this problem only based on diffusion knowledge, from static assumption requiring time stamp or without infection time to dynamic inference. As a step further towards using prior knowledge, some works employ in-degree distribution for nodes, and measurements such as pathways, network properties, and information about the links or nodes.

**Diffusion prediction:** How to identify the future infected nodes in a cascade sequence by observing the incomplete and primary part of the cascade? In specific, the LSTM architectures is used for predicting the next node in a cascade with the help of a complete network structure, estimating the next node or finding the next infection time with the ranking of nodes as the next infection step.

**Diffusion modeling:** This is a task for modeling the pattern and path of cascade over a network. Use EM algorithm to predict propagation with discrete time Independent Cascade (IC) model.

**Missing Detection:** When a complete graph of the network is available, but we have access to the incompetence information propagation, then discovering the source of diffusion is a critical issue. NetFill finds the missing infected nodes and source of diffusion by observing incomplete and noisy cascades with proposing a model based on a Minimum Description Length (MDL).

**Mutual impact of structure and diffusion:** There have been several works on analyzing the dynamic-on (diffusion) and dynamic-off (structure) network. Diffusion affects network evolution, and the network alteration influences the life cycle of diffusion. Therefore, dynamics on and off, have effect on each other. proposes a time-delayed model for new link formation depending on pre-existing links over time. finds new links by classical link prediction methods, then apply diffusion process on new networks and evaluate and analyze the structural and diffusion processes of evolved networks.

To the best of our knowledge, none of the existing research incorporates the simultaneous joint structure and diffusion data to solve the problem of missing data, although they try to recover the missing parts of both, separately.

3 **The Problem Statement**

In this section, we introduce the notations and symbols that will be used in the paper. Then, the problem of joint inference of structure and diffusion networks from partially observed data is clearly stated.

### 3.1 Notations and Definitions

We model a static social network with graph \( G = (V, E) \) where \( V \) represents \( N \) nodes (users of network), \( N = |V| \) and \( E \) indicates the set of edges between nodes. Link \( E_{ij} \) is formed from the relationship of node \( i \) to node \( j \). These edges represent directed and unweighted interactions \( (E_{ij} \neq E_{ji}) \) without considering the self links. Suppose \( G \in \{0,1\}^{N \times N} \) is the corresponding asymmetric adjacency matrix of \( G \). In addition to the network structure, we have a set of information diffusion cascades \( C \) among the users represented by a matrix \( C \in \mathbb{R}^{N \times M} \). By denoting the overall spread of each information as cascade \( C_{ij} \), the element \( C(t_1, c_j) = t_{ij} \) represents that cascade \( c_j \) reached the user \( v_i \) and infected it at hit time \( t_{ij} \in [0,T] \cup \infty \), where \( \infty \) used for users that are not infected by \( c_j \) during the observation window \([0,T] \). We also assume that hit time is set to zero at the beginning of each cascade, and the cascade cannot infect each node more than once during its lifetime.

In this paper, matrices, vectors, and scalars are identified by uppercase bold-faced \((X \in \mathbb{R}^{p \times q})\), lowercase bold-faced \((x \in \mathbb{R}^{p \times 1})\) and normal lowercase \((y)\) letters, respectively. By convention, here, all vectors are column-based, and \( x_i \) is the i-th element of vector \( x \). In a similar manner, \( X_{ij} \) is the j-th column, \( X_i \) is the i-th row of matrix \( X \), and \( X_{ij} \) is the entry in the i-th row and j-th column of \( X \). The \( m \times m \) identity matrix is denoted by \( I_m \). \( X_T \in \mathbb{R}^{q \times p} \) returns the transpose of a matrix and \( vec(X) \in \mathbb{R}^{pq \times 1} \) is the linear operator flattening all the columns of the matrix to a column-based vector. For matrices \( X \in \mathbb{R}^{p \times q} \) and \( Y \in \mathbb{R}^{r \times s} \), \((X \otimes Y) \in \mathbb{R}^{pq \times rs} \) is referred to the Kronecker product of two matrices. If \( p = r, and q = s \) then \((X \circ Y) \in \mathbb{R}^{p \times q} \) denotes the Hadamard product (elementwise).

**Definition 3.1.** Partially observed structure network: Since the links of network graph \( G \) is not fully observable, the available structure of a network is a sub-graph
\[ \mathcal{G} = (V, \mathcal{E}) \subseteq \mathbb{G}. \] Let \( \hat{\mathcal{G}} \) be a \( N \times N \) binary matrix where
the set of one value entries \( \Omega^+ = \{(i, j) : \hat{\mathcal{G}}_{ij} = 1\} \) denotes exiting observed links of the network. \( \Omega^- \) represents the missing links while it may not surely mean to imply that there is no link, but for each entry \((i, j) \in \Omega^-\), we may have \( \mathcal{G}_{ij} = 0 \) or \( \mathcal{G}_{ij} = 1 \) in the oracle network. Therefore, Matrix \( \mathbf{\Omega} \in \{0, 1\}^{N \times N} \) is an indicator matrix that 1 is in the location of \((i, j) \in \Omega^+\) and 0 in the location of \((i, j) \in \Omega^-\).

**Definition 3.2.** Partially observed information diffusion:
A piece of information \( j \) propagates over the structure network, by transmitting through the links from an infected node to a uninfected node. Each cascade can be expressed as a \( N \times 1 \) vector \( \mathbf{C}_j \). Since a contagion does not reach all \( N \) observed users of the network, we face with a sparse vector for each cascade. Let \( \check{\mathbf{C}}_j \) be the observed vector for cascade \( j \), which contains only a subset of the infected users with their hit times. Formally, \( \Gamma^+ = \{(i) : \check{\mathbf{C}}_j \in \mathbb{R}^\times \wedge \check{\mathbf{C}}_j = \mathbf{C}_j\} \) is a set of indices of the observed entries. The state of other nodes of the network \( \Gamma^- = \{\forall \setminus \Gamma^+\} \) is hidden to us, while in the ground-truth of the diffusion process they may be infected or legitimate uninfected nodes. By aggregating all the observed vectors of \( M \) different cascades in a matrix \( \check{\mathbf{C}} \), the mask matrix for partially observable knowledge of diffusion can be represented with \( \Gamma \in \{0, 1\}^{N \times M} \) which is a mapping from the collection of \( M \) vector indices \((\Gamma^+ _{j = 1, \ldots, M} \) and \( \Gamma^+ _{j = 1, \ldots, M} \) to a matrix space.

### 3.2 Relation between Diffusion Behavior and Network Structure

As shown in Figure 2, a network structure includes a set of users and social links between them, while the information propagates among these users. A user would receive a piece of information if one of her/his friends (a user that he has a link to him) had posted or reposted it. Therefore, a node infection can be a sign of how the nodes interact with each other in the network structure. However, the links between users can impact the diffusion process. In conclusion, the diffusion process can reflect and also drive the structure of the network, which demonstrates that the two matrices \( \mathcal{G} \) and \( \mathbf{C} \) are correlated with each other. We will model this correlation property by assuming that \( \mathcal{G} \) and \( \mathbf{C} \) are sharing the same latent factors by employing a coupled matrix factorization.

### 3.3 Problem Statement

Given the partially observed network structure matrix \( \hat{\mathcal{G}} \) and information diffusion matrix \( \hat{\mathbf{C}} \), our goal is to recover the non-observed links of a network by estimating matrix \( \hat{\mathcal{G}} \) and obtain the approximated hit time of unobserved users of the network who are interacting hiddenly in diffusion processes with discovering matrix \( \hat{\mathbf{C}} \).

### 3.4 Matrix Factorization

Since our goal is recovering the hidden entries of two partially observed matrices, it can be modeled like a kind of matrix completion problem. Matrix factorization is a prevalent and effective technique for matrix completion problem by approximating a given matrix \( \mathbf{A} \in \mathbb{R}^{m \times n} \) as a product of two low-rank latent factor matrices \( \mathbf{W} \in \mathbb{R}^{m \times r} \) and \( \mathbf{H} \in \mathbb{R}^{r \times n} \) with constraint on \( r \leq \min\{m, n\} \), such that \( \mathbf{A} \approx \mathbf{WH} \). The latent factors \( \mathbf{W} \) and \( \mathbf{H} \) are \( r \)-Dimension representation matrices and can be interpreted as embedding for rows and columns of \( \mathbf{A} \). These factors can be learned even if \( \mathbf{A} \) is partially observed by minimizing the reconstruction error for the observed entries in order to recover the full \( \mathbf{A} \).

### 4 THE PROPOSED METHOD

In this section, we present the proposed model framework and how to infer the latent factor matrices. Table 2 summarizes the notations that we use in the proposed method.

#### 4.1 Model Framework

The cascade matrix \( \mathbf{C}_i \) can be modeled as:

\[
\mathbf{C}_{ij} = \mathbf{X}_i^T \mathbf{Y}_j + \epsilon_{ij} \quad \epsilon_{ij} \sim \mathcal{N}(0, \sigma^2_{\epsilon_{ij}}) \quad (1)
\]

where \( \mathbf{X} \in \mathbb{R}^{D \times N} \) and \( \mathbf{Y} \in \mathbb{R}^{D \times M} \) are low-rank matrices representing the user and cascade latent features, and \( \epsilon_{ij} \) denotes the residual noise sampled independently from zero-mean Gaussian distribution with variance \( \sigma^2_{\epsilon_{ij}} \). Therefore, the conditional distribution over all the observed infected nodes is:

\[
P(\mathbf{C}|\mathbf{X}, \mathbf{Y}, \sigma^2_{\epsilon_{ij}}) = \prod_{i=1}^{N} \prod_{j=1}^{M} \mathcal{N}(\mathbf{C}_{ij}|\mathbf{X}_i^T \mathbf{Y}_j, \sigma^2_{\epsilon_{ij}})^{\Gamma_{ij}}(2)
\]
Therefore, we resort to a deterministic model. The hyper-factor idea for joint inference is incorporating the shared latent and structure of the network called “DiffStru”. Our basic model can handle these empty rows of matrices by integrating her/his diffusion behavior is available, and hence, the corresponding rows of latent factors will be empty. However, we can handle these empty rows of matrices by integrating the side information as prior knowledge for capturing the correlation between users or cascades. Here, a zero-mean multivariate Gaussian distribution is employed as a conjugate prior:

\[ Y_d \sim N_M (0, W_Y), X_d \sim N_N (0, W_X), U_d \sim N_N (0, W_U) \]

We exploit the diffusion and topological metrics as prior distributions for each row of the latent matrices \( Y_d, U_d, \) and \( X_d, \) with covariance matrices \( W_Y, W_U, \) and \( W_X, \) respectively. Each element of full covariance matrices \( W_Y \in \mathbb{R}^{N \times N} \) or \( W_X \in \mathbb{R}^{N \times N} \) in these distributions capture the relationship between the pair of cascades (users) and also the correlation between different features of a cascade (user). These covariance matrices will apply both the covariances between rows and columns of matrices in the priors [35]. We will show later in Section 5.4 how to initialize these covariances.

In the general formulation based on the graphical model in Figure 4, the joint distribution over the observed, latent, and auxiliary random variables given the hyper-parameters \( \Theta = \{ W_Y, W_X, W_U, \alpha_1, \alpha_2, \sigma_C^2, \sigma_R^2, \Pi \} \) is given by:

\[
P(X, U, Y, R, \Xi, \mu_\Xi, G, C|\Theta) = \prod_{i=1}^{M} \left( \prod_{j=1}^{N} (N(R_{ij}|X_i^T U_j, \sigma_R^2)) \times [N(C_{ij}|X_i^T Y_j, \sigma_C^2)]^{\delta_{ij}} \times [\text{Bernoulli}(G_{ij}|f(R_{ij}))]^{\Xi_{ij}} \times \text{Bernoulli}(\Xi_{ij}|\mu_\Xi)\text{Beta}(\mu_\Xi|\alpha_1, \alpha_2) \right) \prod_{d=1}^{D} (N_d(Y_d|0, W_Y) \times N_N(X_d|0, W_X) \times N_N(U_d|0, W_U))
\]

We should learn the posterior distribution to estimate the unobserved data by employing a Bayesian approach. Using the Bayes rule, the posterior becomes an intractable integral. For approximating these integrals, the MCMC algorithm is a common approximation approach that tries to obtain sufficient number of samples of the target distribution for a dependable inference [36].

**Algorithm 1:** Generative process for DiffStru graphical model

1. \( \mu_\Xi \sim \text{Beta}(\alpha_1, \alpha_2) \)
2. \( Y_d \sim N_M (0, W_Y), \forall d = 1, ..., D \)
3. \( X_d \sim N_N (0, W_X), \forall d = 1, ..., D \)
4. \( U_d \sim N_N (0, W_U), \forall d = 1, ..., D \)
5. foreach user \( i = 1, ..., N \) do
6.   foreach user \( j = 1, ..., N \) do
7.     \( \Xi_{ij} \sim \text{Bernoulli}(\mu_\Xi) \)
8.     \( R_{ij} \sim N(X_i^T U_j, \sigma_R^2) \)
9.     \( G_{ij} \sim \text{Bernoulli}(f(R_{ij})) \) \( + (1 - \Xi_{ij})(\delta_0) \)
10. end
11. foreach cascade \( r = 1, ..., M \) do
12.   \( C_{ir} \sim \Pi_{ir}(N(X_i^T Y_{ir}, \sigma_C^2)) \)
13. end
14. end
4.2 Model Inference

There are various ways to sample a distribution. In this case, we use the Gibbs method as a MCMC technique, by iterative sampling one variable from a conditional distribution with fixing the remaining variables. In each iteration, all random variables are updated based on the previous value of others. We randomly initialize the values of random variables. To minimize the effect of random initialization, we should drop the first early samples, which is known as choosing the sufficient burn-in period. Also, thinning the chain is needed to avoid bias estimation to correlated samples and reduce the cost of processing and storing. Thinning is done by not considering the correlated samples and averaging over every k-th iteration. The proof of the following equations can be found in the appendix for the sake of space and readability.

**Sampling Y:** For sampling from variable Y, we estimate conditional distribution of Y using the joint distribution:

\[
P(Y|\cdot) \propto P(C|\Pi_{ij}, X^T, Y, \sigma_2^2)P(Y|0, W_Y)
\]

\[
= \prod_{i=1}^{N} \prod_{j=1}^{M} [N(X_i^T, Y_j, \sigma_2^2)]^{\Pi_{ij}} \times \prod_{d=1}^{D} [N_M(Y_d|0, W_Y)]
\]

**Properties 4.1. Matrix Normal Distribution:** A r \times t matrix \( Z \) follows matrix normal distribution with mean matrix \( T \) and covariance matrix \( \Sigma_Z = B^t \otimes A \times r \times t \): \( Z \sim MN_{tr}(T, A, B) \), where \( A \) and \( B \) are positive-definite real matrices. Then the probability density function is:

\[
\exp \left( -\frac{1}{2} tr[B^{-1}(Z-T)^T A^{-1}(Z-T)] \right)
\]

**vec** operator it also has the form:

\[
vec(Z) \sim MN_{tr}(vec(T), \Sigma_Z)
\]

Where \( tr \) is the matrix dimension with \( t \) rows and \( r \) columns.

**Properties 4.2. Transpose transform of Matrix Normal:** If matrix \( Z \) follows matrix normal distribution \( Z \sim MN_{tr}(S, A, B) \), then the transpose of \( Z \) is also a matrix normal distribution with the same covariance parameters and transposed mean matrix: \( Z^T \sim MN_{tr}(S^T, A, B) \).

In order to find the posterior for sampling \( Y \), first we re-write the above relation in the form of a normal matrix.

Due to conjugacy with using properties (4.1) and (4.2), the posterior \( P(Y|\cdot) \) is also Gaussian by using the vec operator:

\[
P(vec(Y)|\cdot) \sim \mathcal{N}_{DM}(vec(Y)|\mu_Y, \Sigma_Y)
\]

Where the covariance \( \Sigma_Y \) and mean \( \mu_Y \) are given by:

\[
\Sigma_Y = \sum_i \left[ (\mathbb{I}_M \otimes X) \odot (\sigma^2_2 vec(vec(Z)^T)) \right] (\mathbb{I}_M \otimes X^T)^T + (\mathbb{I}_D \otimes W_{Y^{-1}})^{-1})^{-1}
\]

\[
\mu_Y = \sum_i \left[ (\mathbb{I}_M \otimes X) \odot (\sigma^2_2 vec(vec(Z)^T)) \right] vec(C)
\]

Note that \( W_2 \) is a fixed hyper-parameter covariance matrix and \( vec(C) \) is the fixed mask matrix. Therefore, the terms \( \sigma^2_2 vec(vec(Z)^T)^2 \) and \( \mathbb{I}_D \otimes W_{Y^{-1}} \) will not be updated in each iteration and we can use them directly as inputs.

**Sampling \( \Xi \):** The conditional posterior density of \( \Xi \) is:

\[
P(\Xi|\cdot) \propto P(G|R, \Xi, \Xi)P(\Xi|\mu)
\]

As discussed in the previous section, \( \Xi_{ij} = 1 \) when \( \Gamma_{ij} = 1 \), and hence:

\[
P(\Xi_{ij}|G_{ij} = 0, \cdot) = [\Xi_{ij}(1 - f(R_{ij})) + (1 - \Xi_{ij})(G_{ij} = 0)] \times \mu_{\Xi_{ij}}^{1 - \Xi_{ij}} \Xi_{ij}^{1 - \Xi_{ij}}
\]

But this value should be inferred for elements when \( \Gamma_{ij} = 0 \). In this case \( P(\Xi_{ij}) \) is a Bernoulli distribution:

\[
P(\Xi_{ij} = 1 | G_{ij} = 0, \cdot) = \text{Bernoulli}(\xi) \quad \xi = \frac{\mu_{\Xi} - \mu_{\Xi} f(R_{ij})}{1 - \mu_{\Xi} f(R_{ij})}
\]

**Sampling \( \mu_{\Xi} \) and \( \mu_{\Xi} \):** Due to the use of conjugate prior for \( \mu_{\Xi} \) and \( \mu_{\Xi} \), their conditional distribution will be a Beta distribution. Equation (15) shows the details for \( \mu_{\Xi} \):

\[
P(\mu_{\Xi} | \cdot) \propto P(\Xi_{\mu_{\Xi}}, \sigma_{\mu_{\Xi}}^2)P(\mu_{\Xi} | \alpha_1, \alpha_2)
\]

\[
= \prod_{i=1}^{N} \prod_{j=1}^{M} \frac{1}{B(\alpha_1, \alpha_2)^{\Xi_{ij}^{\alpha_1 - 1}(1 - \Xi_{ij})^{\alpha_2 - 1}}}
\]

\[
\sim \text{Beta} (\mu_{\Xi}|\alpha_1, \alpha_2 + N^2 - \sum_{i,j} \Xi_{ij})
\]

**Sampling U:** Given the other latent variables, the posterior of \( U \) is:

\[
P(U|\cdot) \propto P(R|X^T, U, \sigma_2^2)P(U|0, W_U)
\]

\[
= \prod_{i=1}^{N} \prod_{j=1}^{M} [N(R_{ij}|U_{ij}, \sigma_2^2)] \times \prod_{d=1}^{D} [N(U_d|0, W_U)]
\]

Therefore, we can sample vec(\( U \)) from the following distribution:

\[
P(\text{vec}(U)|\cdot) \sim \mathcal{N}_{DN}(\text{vec}(U)|\mu_U, \Sigma_U)
\]

Where the covariance \( \Sigma_U \) and mean \( \mu_U \) are given by:

\[
\Sigma_U = \sum_{U} \left[ [(\sigma^2_2 \mathbb{I}_N) \otimes (X^T X)] + (\mathbb{I}_D \otimes W_{U^{-1}})^{-1})^{-1}
\]

\[
\mu_U = \sum_{U} \left[ [(\sigma^2_2 \mathbb{I}_N) \otimes (X)] \text{vec}(R) \right]
\]
Note that $\sigma^2_N I_N$ and $I_D \otimes W_U^{-1}$ are fixed input terms. There is a problem in sampling the remaining variables of the joint distribution (7), because of the logistic likelihood function which is not conjugate with other Gaussian terms. Hence, we utilize the Polya-Gamma latent variables by adding an auxiliary random variable $\Lambda$ for approximating the logistic likelihood with a Gaussian distribution that can easily multiply with prior normal distributions.

$$P(G_{ij}|R_{ij}, E_{ij} = 1) = f(R_{ij}) = \frac{1}{1 + e^{-\Lambda_{ij}}} = \frac{e^{R_{ij}}}{e^{R_{ij}} + 1}$$

$$P(A_{ij}) \sim PG(\Lambda_{ij}|1, 0)$$

where, $P(A_{ij}) \sim PG(\Lambda_{ij}|1, 0)$. By conditioning (19) on auxiliary variable, we obtain:

$$P(G_{ij}|R_{ij}, E_{ij} = 1) = 1, A_{ij}) \times \exp((G_{ij} - 0.5)R_{ij} - \frac{1}{2}A_{ij}R_{ij}^2)$$

**Sampling $\Lambda$:** The sampling of $\Lambda$ is given by:

$$P(\Lambda_{ij}|R_{ij}, E_{ij} = 1, A_{ij})P(\Lambda_{ij}) \sim PG(\Lambda_{ij}|1, 0)$$

**Sampling $X$:** The conditional distribution of user latent features using the joint distribution of (7) is:

$$P(X|\Lambda, \Xi) = \mathcal{P}(\mathbb{C}, \mathbb{X}) = \mathcal{P}(\mathbb{X}^T U, \sigma^2_R)P(X|0, W_X)$$

$$= \prod_{i=1}^{N} \prod_{j=1}^{M} \mathcal{N}(C_{ij}|X_{ij}^T Y_{ij}, \sigma^2_R) \Pi_{ij}$$

$$\times \prod_{i=1}^{N} \prod_{j=1}^{M} \mathcal{N}(R_{ij}|X_{ij}^T U_{ij}, \sigma^2_R) \times \prod_{d=1}^{D} \mathcal{N}(X_d|0, W_X)$$

The posterior in (21) is also Gaussian by using the vec operator. For simplicity, we obtain the parameters of the transpose of $X$ which has also a normal distribution:

$$P(\text{vec}(X^T)|\cdot) \sim \mathcal{N}_{DM}(\text{vec}(X^T)|\mu X^T, \Sigma_X)$$

where the covariance $\Sigma_X$ and mean $\mu X^T$ are given by:

$$\Sigma_X = \sum_{X} \left[ \left( (Y \otimes I_N) \circ (\sigma^2_C(\text{vec}(\mathbb{C}))^T) \right) (Y^T \otimes I_N) \right]$$

$$\mu_X = \sum_{X} \left[ \left( (Y \otimes I_N) \circ (\sigma^2_C(\text{vec}(\mathbb{C}))^T) \right) \text{vec}(\mathbb{C}) \right]$$

**Sampling $R$:** The sampling of $R$ is given by:

$$P(R_{ij}|\cdot) \sim \mathcal{N}(R_{ij}|\mu_{R_{ij}}, 1)$$

where the mean $\mu_{R_{ij}}$ is given by:

$$\mu_{R} = \mathbb{E}_{ij}(G_{ij} - 0.5)\sigma^2_R + \mathbb{X}_T U_{ij} \mathbb{E}_{ij} \Lambda_{ij} \sigma^2_R + 1$$

The DiffStru algorithms can be found in Algorithm 2.

**Algorithm 2: DiffStru Gibbs Sampler.**

**Input:** Matrix Cascade $C$, Matrix Graph $G$, $W_Y^{-1}$, $W_X^{-1}$, $U T^{-1}, \alpha_1, \alpha_2, \sigma^2_C, \sigma^2_R$, max iteration $T$, burn in $b$, thinning $k$, and latent factor $D$

**Output:** $X, U, Y, \Xi$

1. $(N, \bar{N}) = \text{shape}(G)$;
2. $(N, M) = \text{shape}(C)$;
3. Initialize $X^D \times N, R^N \times N, Y^D \times M, U^D \times N, \Lambda^N \times N, \mu^N \times N$ randomly;
4. Initialize binary $\Xi^N \times N$ and $\Pi^N \times M$ from $G$ and $C$;
5. for $t = 1, ..., T$
6. Sample $R_{ij}$ from (14);
7. Sample each element of $\lambda_{ij}$ from (19);
8. Sample $X^T$ from (22);
9. Sample $U^T$ from (17);
10. Sample $Y^T$ from (9);
11. Sample each element of $A_{ij}$ from (20);
12. Sample $\mu^T$ from (19);
13. end
14. $X = \frac{k}{T - b} \sum_{i=1}^{T-b} X_{ij}^{b+ki}$, $Y = \frac{k}{T - b} \sum_{i=1}^{T-b} Y_{ij}^{b+ki}$,
15. $U = \frac{k}{T - b} \sum_{i=1}^{T-b} U_{ij}^{b+ki}$, $\Xi = \frac{k}{T - b} \sum_{i=1}^{T-b} \Xi_{ij}^{b+ki}$

### 4.3 Predicting Missing Data

By learning the latent matrices $X, Y, U$, and $\xi$, $\hat{G} = \sigma(X^T U)$ can be estimated. For the missing entry of $\hat{G}_{ij}$, $\hat{G}_{ij}$ is given by:

$$\hat{G}_{ij} = \begin{cases} 0 & \text{if } (\xi_{ij} = 0) \text{ and } (\sigma(X^T U_{ij}) \leq \delta_C) \\ 1 & \text{else} \end{cases}$$

where $\delta_C$ is a threshold for distinguishing links and non links. To approximate the missing values of $\hat{G}_{ij}$, first we define the infection probability of the node $i$ in the cascade $j$ with $P_{ij}$, and if $P_{ij} > \delta_C$ then $C_{ij}$ is estimated by $X_{ij}^T Y_{i,j}$ with mapping the negative and the out of interval $[0, T]$ values to uninfected state:

$$z = X_{ij}^T Y_{i,j}, \hat{G}_{ij} = \begin{cases} 0 & \text{if } (P_{ij} > \delta_C) \text{ and } (z < 0 \text{ or } z > T) \\ 1 & \text{else} \end{cases}$$

The infection probability matrix is defined as $P = \Pi^T A$, where $\Pi$ is the mask diffusion matrix that we have already defined. $A$ is a $N \times N$ infection transfer matrix, that each
of its \((i,j)\) elements indicates the probability of infection propagation from node \(i\) to node \(j\) as:

\[
A_{i\rightarrow j} = \frac{\left| \hat{C}(i,:) \cap \hat{C}(j,:) \right|}{\left| \hat{C}(i,:) \cup \hat{C}(j,:) \right|}
\]  

(27)

From the diffusion observations we obtain the probability that the simultaneous concurrence of two nodes \(i\) and \(j\) in cascades is due to the infection transmission from \(i\) to \(j\).

5 NUMERICAL AND SIMULATION RESULTS

In this section, we first describe the synthetic and real datasets with related evaluation metrics. Then, we analyze the performance and efficacy of DiffStru. Since there is no related work that addresses joint completion of the missing parts of diffusion and structure, we separately compare DiffStru against different methods in link prediction and cascade prediction.

**Structure part competitors:** We chose Adamic Adar (AA) \([38]\), Resource Allocation (RA) \([39]\), and Common Neighbor (CN) \([40]\) as classical link prediction methods as well as the recent fusion matrix factorization method (FPMF) \([9]\). The classical approaches calculate the similarity weight of any unlinked pairs of nodes. By sorting the weights in ascending order and cutting the top pairs, unobserved links can be obtained. FPMF is a matrix factorization method that fuse some asymmetric and symmetric topological metrics with adjacent structure matrix to learn the latent factor of nodes with gradient methods.

**Cascade part competitors:** There is no work for mining the omitted infected nodes in the diffusion category with estimating their infection time. To compare this part of DiffStru, we chose a cascade prediction technique called DeepDiffuse \([27]\). DeepDiffuse tries to predict the next node in a cascade by ranking the possible candidates with estimating a single time. However, this method does not pay attention to missing data, and it aims to predict the next step of the cascade sequence by assuming that the observed steps are complete. Therefore, to compare DeepDiffuse with DiffStru, we should input the cascade sequence up to any missing step point to the model in order to obtain the prediction. Then we can compare the error of predicted next infection time. A baseline, we fit a polynomial regression model (with degree one and two, named as Reg-1 and Reg-2) on a sorted cascade sequence. Then for any missing node \(k\), which is infected after node \(i\) and before node \(j\), we find the related time from the learned regression model using the mean indexes of node \(i\) and \(j\), in the sequence. It is clear that for both DeepDiffuse and baseline methods, we provide more prior knowledge in comparison with DiffStru. Both of these methods know position of two nodes in a cascade where the missing was happened. Therefore, computing the missed infection time is approximated relative to the time of previous node. However, DiffStru does not utilize this information.

5.1 Description of the Datasets

**Synthetic:** Independent of the proposed model, we generated synthetic directed graphs by resembling real social networks. The LFR (Lancichinetti-Fortunato-Radicchi) network

1. All codes of DiffStru can be downloaded at:

is a benchmark having interesting real-world features such as community structure and power-law degree distribution \([41]\). We generated artificial LFR networks with mixing parameter 0.1, degree sequence exponent 2, community size distribution exponent 1, and zero number of overlapping nodes. Then on each graph, we simulated independent cascades with different lengths using the method described in \([15]\). The transmission model was exponentially distributed with parameters: alpha=1, mixture of exponential=1 and beta=0.5. The details of generated datasets are listed in Table 3.

**Real world:** We utilized two real datasets: A Twitter dataset during October 2010 with a network of followers and time sequence of retweeting between the users \([42]\). Also, a Memetracker dataset with memes propagation between websites during April 2009 with a connection graph \([43]\). As stated earlier, data loss in real-world datasets is unavoidable. Thus, creating a test scenario on these datasets will cause an extreme lack of information, and no proper ground-truth exists for evaluation. Therefore, we consider a dense part of these networks (by choosing a big community) to test and perform data deletion scenarios. The final statistics of these datasets is listed in Table 3.

| Dataset    | #nodes | #links | density | #cascades | activites |
|------------|--------|--------|---------|-----------|-----------|
| LFR100     | 100    | 2021   | 0.204   | 200       | 15684     |
| LFR400     | 400    | 19987  | 0.123   | 600       | 60600     |
| Twitter    | 203    | 6642   | 0.162   | 214       | 214       |
| Memetracker| 416    | 41404  | 0.239   | 287       | 4828      |

5.2 Evaluation Metrics

To evaluate the results of DiffStru, we compared its performance from structural and diffusion points of view. The observed \(\hat{G}\) and \(\hat{C}\) are the training sets, and we treat them as known data, while the missing parts of \(G\), \(\hat{G}\) and \(C\), \(\hat{C}\) are testing sets, and no knowledge about these sets is used in learning our model. If the output of models are represented with \(\hat{G}\) and \(\hat{C}\), then we evaluate the reconstruction of estimated matrices, and accuracy of different aspects with the following metrics.

**SRE:** With Signal Reconstruction Error (SRE) \([44]\), we can evaluate the success rate in reconstructing the ground-truth matrix - the higher this metric, the lower the recovery noise ratio to the original matrix. For the ground-truth matrix \(Z\), if the estimated matrix is \(\hat{Z}\), SRE metric is approximated as \([44]\):

\[
SRE = \frac{\mathbb{E} \left[ \|Z\|_2^2 \right]}{\mathbb{E} \left[ \|Z - \hat{Z}\|_2^2 \right]} 
\]

(28)

**AUC:** Since we face an unbiased binary classification in reconstructing the structural information (matrix \(G\)), the area under the receiver operating characteristic curve is a useful metric for evaluation, which is independent of the threshold setting. AUC measures the probability of randomly selecting linked and unlinked pairs of nodes and checking if the probability of linked pairs is higher than unlinked pairs.
The higher AUC value indicates the better performance of model in classifying the two classes.

\[
AUC = \left( \sum_{(a, b), (c, d) \in (G, \hat{G})} \left[ I(\hat{G}(a, b) > \hat{G}(c, d)) + 0.5I(\hat{G}(a, b) = \hat{G}(c, d)) \right] \times I(G(a, b) = 1, G(c, d) = 0) \right) / \left( \sum_{(a, b), (c, d) \in (G, \hat{G})} I(G(a, b) = 1, G(c, d) = 0) \right)
\]

(29)

**Precision, Recall, and F-measure:** By applying threshold on probabilities of matrix \( \hat{G} \), we can map the elements to zero and one values. The precision and recall are defined below, and F-measure is the weighted harmonic mean of recall and precision.

\[
pre = \frac{\hat{G} = 0 \land G = 1 \land \hat{G} = 1}{G = 1 \land \hat{G} = 0}, \quad rec = \frac{\hat{G} = 0 \land G = 1 \land \hat{G} = 1}{G = 1 \land \hat{G} = 0}
\]

(30)

**Accuracy:** Is a metric for measuring the portion of correctly classified pairs:

\[
ACC = \frac{(G = 0 \land G = 1 \land \hat{G} = 1) + (G = 0 \land G = 0)}{G = 0}
\]

(31)

**MCC:** The Matthews Correlation Coefficient metric [15] is suitable for measuring quality of binary ill-balanced classification which utilizes the ratios of true positives, true negatives, false positives, and false negatives. When the value of this metric is near 1, the estimated value is closer to the reality, while the value closer to 0, shows the estimator prediction is close to random, and the -1 value indicates a complete disagreement.

**MAP@K:** The Mean Average Precision is utilized for measuring the performance of DeepDiffuse [27]. If we have "A" number of missing activities, the ground-truth of these test sample has one value, while DeepDiffuse outputs an ordered list of nodes. MAP@K means averaging overall prediction of "A" samples based on the @K top first of the algorithm output list. We tested K = 10, 50, 100 here.

\[
MAP@K = \frac{\sum_{i=1}^{A} I(r_i \leq K)(\frac{1}{r_i})}{A}
\]

(32)

**RMSE:** We measure the Root Mean Square Error (RSME) between C and \( \hat{C} \) for timestamp of observed training data, and test unobserved nodes. There is no fixed threshold limit for RMSE and the lower value indicates better fit.

### 5.3 Experiments Setting

The first step towards setting inputs of experiments is simulating the missing information in all datasets. Missing data can occur at random events, which leads to an unbiased analysis. However, in some real cases, data missing may happen because of sampling technique or some limitation of APIs that are not random. For example, we cannot gather all the connections of a high degree node that direct us to missing data. Since we do not have any assumption about deletion in the proposed method, we ran different experiments for two types of missing data.

**Random missing:** We eliminate data with uniform distributions to simulate random missingness. For this purpose, first choose a remove rate \( \theta \in [0, 1] \), then for each link of structure or activity in the diffusion, a random value \( \tau \) is generated from uniform distribution on \([0, 1]\), and if \( \tau < \theta \) we remove that data.

**Non-random missing:** For a non-random scenario we randomly remove the links between nodes whose outdegree is higher than five. Besides, we calculate the activities of each node, and activity removal is done randomly over the set of nodes that have more than five participants.

Accordingly, in the next section, results are for two missingness scenarios: random and non-random. By default, runs were over 1000 iterations with burn-in of 900 and thinning of 1. If there is a change in the setting of any scenario, we have listed the new settings.

### 5.4 Hyper-parameters Setting

The link prediction between any two nodes is correlated with the structural properties of nodes so that the more similar structural relations of two nodes will lead to more likelihood of link existence between them. Furthermore, the cascades that have infected more similar nodes are more alike in their diffusion path. Suppose that \( \Theta^\Sigma \) is a measure of the structural similarity between two nodes \( i \) and \( j \) in a network, then the larger value of \( \Theta^\Sigma \) indicates the lower distance between \( X_i \) and \( X_j \). Therefore, for any pair of nodes we look for the minimal value of the following equation:

\[
\sum_{i=1}^{N} \sum_{j=1}^{N} \Theta^\Sigma_{ij} \| X_i - X_j \|_2^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{d=1}^{D} \Theta^\Sigma_{ij} (X_{di} - X_{dj})^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{d=1}^{D} (\Theta^\Sigma_{ij} X_{di} - \Theta^\Sigma_{ij} X_{dj}) = \sum_{d=1}^{D} X_d^T (\zeta^\Sigma - \Theta^\Sigma) X_d.
\]

(33)

where \( L^\Sigma = (\zeta^\Sigma - \Theta^\Sigma) \) is a Laplacian matrix where \( \zeta^\Sigma_{ii} = \sum_{j=1}^{N} \Theta^\Sigma_{ij} \) is a diagonal matrix. This property of \( X \) is equivalent to multivariate Gaussian distribution with zero mean and inverse covariance matrix \( W_X^{-1} = L^\Sigma \). Similarly, we can model \( U \) and \( Y \) with zero mean multivariate Gaussian densities with \( W_U^{-1} = L^U \) and \( W_Y^{-1} = L^Y \) as inverse covariances.

To initialize the covariance matrix of hyper-parameters, we can set it to an identity matrix without modeling the correlation of nodes and cascades (independent priors) or considering the relation of these components (correlated priors). For dependent priors, we use the similarity of nodes as the number of common neighbors in the partially observed structure network for \( \Theta^U \) and \( \Theta^Y \), while the similarity of cascades is defined by the \( ij \) entry of \( \Theta^Y \), which is the number of common nodes in \( i \)-th and \( j \)-th cascades.

Other hyper-parameters are set as, in Equation (26) where \( \delta_G = 0.5 \), and \( \delta_C \) in Equation (25) is set as the mean value of all \( P \) elements. Moreover, \( \sigma_C^2 = 1, \sigma^2 = 1, D = 8, \alpha_1 = 0.2, \) and \( \alpha_2 = 0.3 \).

### 5.5 Model Analysis

In this section, we verify the impact of different parameters on the performance of DiffStru and focus on its ability to
TABLE 4: Synthetic LFR datasets with 50 nodes and 50 cascades over them for analyzing the performance of model in term of network density. Following statistics are for ground-truth (Gt) and observation (Ob). The columns from left to right are: number of links in ground-truth structure, number of communities in the ground-truth, density of underlying network, density of observed network, number of activities in the oracle cascades, and number of observed activities.

| Data | Structure | Diffusion |
|------|-----------|-----------|
| #links | #com | den (Gt) | den (Ob) | #act (Gt) | #act (Ob) |
| 1 | 740 | 2 | 0.30 | 0.18 | 2050 | 1450 |
| 2 | 461 | 2 | 0.19 | 0.11 | 1605 | 1127 |
| 3 | 197 | 8 | 0.07 | 0.04 | 444 | 335 |
| 4 | 98 | 16 | 0.04 | 0.02 | 207 | 193 |

solve problems in various situations. The data used here are synthetic LFR networks with parameters that are shown in Tables 3 and 4. Some experiments are intended to answer the following questions.

(1) What is the best size of the latent space (hyper-parameter D)? Obviously, learning the latent factors of the model in the smaller dimension will reduce the complexity of inference calculations. On the other hand, the model focuses on simultaneously completing the structure and diffusion matrices. While the measurement metrics for completing each of these matrices are different, we scan for a lower dimension where the best results can be achieved in both structure ad diffusion spaces. Because of random initialization, we ran the model multiple times and reported the mean and standard deviation of metrics, as shown in Figure 6. Here, the different values of latent factor are analyzed, and D = 8 is the best choice with higher mean value and lower standard deviation in terms of AUC, precision, and F-measure for mining missing links, as well as, lower mean and standard deviation for RMSE in finding missing infected and uninfected activities.

Fig. 6: Comparing structural and diffusion metrics against different values of latent space in LFR100 dataset.

(2) How does the density of the ground-truth network structure affects the inference of missing information? We simulated four different LFR datasets with 50 nodes and 50 generated cascades with properties shown in Table 4 to have various densities. Then in each dataset, we randomly removed the activities in cascades with 0.3 rates and the remaining 60% of links as observed data. The results are reported based on an average of five different training sets along with their standard deviations. The setting of sampling was 2000 number of iterations, burn-in of 1500, and thinning of 5. Figure 7a shows the structural metrics against different densities for the ground-truth graph. As illustrated, by increasing the amount of information, the performance of the method is increased. As shown in Figure 7b, more links lead to better cascade estimation. Moreover, we achieve acceptable results only by observing approximately half of the data in structure and diffusion layers.

(3) How is the performance of DiffStru even if there is no activity observation for a subset of users (e.g., users with private accounts)? In the LFR50 dataset, we used the same removal setting as the previous analysis mentioned in question (2), and increased the amount of data deletion by removing the whole information of five rows of diffusion matrix to simulate the private user profiles. In addition to accessing half of the users activities, we did not have

Fig. 7: Impact of ground-truth density in LFR100 dataset. Since only 60% of links are observed, the values of observed network density are {0.023, 0.044, 0.11, 0.18} for horizontal axis, respectively.
any data about the activities of five nodes of network. Then, two settings of the proposed method (correlated and independent priors) were tested. Figure 8 demonstrates the result of DiffStru when the hyper-parameter covariance matrix of prior distribution is initialized with correlated side information, and when the priors are independent by utilizing an identity matrix. We found that the dependent prior can increase the performance of DiffStru for both structure and diffusion information, especially when there are empty rows in the cascade matrix.

![Fig. 8: Effect of using correlated values for initializing the covariance matrix of prior distributions in LFR50 dataset.](image)

(4) Can the learned latent factors be used for classification problems? Each node and cascade can be represented with an embedded $D \times 1$ vector. LFR100 and LFR400 datasets have four and six ground-truth embedded community structures, respectively. We visualized the network nodes from the learned matrix $X$ and $U$ with the color of their communities as shown in Figures 9a, 9c, 9b, and 9d, by using the PCA and t-SNE methods [46]. As illustrated in the figure, the embedded learned features of nodes are exactly separated in space according to their community labels, while we did not consider any assumption on the community structure in DiffStru. Despite node classification, we do not have any ground-truth for comparing the cascades classifications, but their embedded vectors based on the learned matrix $Y$ is shown in Figures 9e and 9f.

(5) What is the impact of the removal rate on the performance of DiffStru? We investigated this issue on the LFR100 dataset. First, keeping the complete diffusion information, we examined the effect of link removal in the output of link mining. In this scenario, there were no test for cascades. The infection times estimator was tested in the training set of cascades. We randomly chose 203 samples of links from the structure network for test, and then in experiments, we decreased the number of observations to find out the impact of link observation on the performance of the method, as shown in Figure 10. Second, using the full observed graph of nodes, we tested the different missing rates in the data of cascades. We chose 1655 activities as test and did the experiments on the various number of observations. Here, we evaluated cascade information, and there were no missing links for testing. In Figure 11, the performance of DiffStru on the three splits of data is represented. RMSE on the training data is almost constant, while it decreases by increasing the number of observations in the test set, including infected and uninfected nodes.

![Fig. 9: Visualization of learned latent feature matrices.](image)

5.6 Comparison

We tested DiffStru against related works on two synthetics and two real datasets. Based on the description in Section 5.3 different missingness patterns were utilized for generating the test scenarios. The Gibbs sampler of DiffStru was run with 5000 iterations, and burn-in of 4500 and thinning of 5. The comparison over the structural network is reflected in Table 6 for random missing, and Table 7 for non-random missing. In both cases, DiffStru outperforms others in different metrics, specially simultaneously in AUC and F-measure, which are essential performance metrics for imbalanced datasets. Analysis from the diffusion perspective is reported in Table 5. Notably, the range of RMSE value does not have a limited ceiling. The remarkable point is DiffStru infers any missing node of a cascade with its infection time, but DeepDiffuse outputs a unique infection time and sort all nodes of the network in order of their probability that is suggested for the next infected node in the cascade sequence. While DiffStru exactly infer the missing node and its timestamp, one should evaluate the precision of DeepDiffuse for discovering the node. Besides, the output of DiffStru is a single node with a real-value time, while DeepDiffuse ranks the nodes. We also had to report the MAP@K metric for DeepDiffuse, while this metric is not
methods is shown in Figure (12). The break-even points of DiffStru is near 0.54, 0.53, 0.94, and 0.79 for LFR100, LFR400, Twitter, and Memetracker datasets, respectively which are all higher than the other methods.

In general, DiffStru performs better than the competing methods in terms of different metrics of Section (5.2). For the diffusion network comparisons, due to the lack of similar work, we had to make unfair comparisons with DeepDiffuse and Regression models for evaluating the RMSE of missing infection times. DeepDiffuse provides a sorted list of nodes instead of suggesting a specific missed node. The RMSE of infection times for the test data, and MAP@K for predicting the name of a node is listed in Table (12). Since the regression models cannot detect identity of the node, the corresponding table cells for MAP@K are filled with ‘x’, and just RMSE is reported for its two versions Reg-1 and Reg-2. On the other hand, DiffStru finds the missing node with its infection time and its RSME is reported in the table. Because of exact inference of DiffStru, its MAP@K is always 100%, and hence we have reported this fact by filling the corresponding cells in the table with ‘. It can be seen that despite the accurate identification of the missing node with DiffStru, its RMSE for infection time is in the appropriate range. In the RMSE comparison, it is important to note that the maximum value of RMSE is not limited to one.

6 CONCLUSION AND DISCUSSION

In this paper, we presented a novel generative model called DiffStru by combining both partially observed structure and diffusion information of a network to efficiently infer the missing data. By fitting suitable distributions and estimating parameters with Gibbs sampling, we embedded the observations in a low-dimensional latent space. DiffStru learns the latent factors during the inference, and hence can be utilized to solve other related network classification problems such as community detection. We conducted several experiments on synthetic and real datasets for measuring the effectiveness of DiffStru. As a future work, instead of the inner product of latent matrices, a deep-learning approach can be utilized for model inference, in order to capture the complex relations of the data. In addition, our generative model can be extended to support more coupling relations.

APPENDIX

1 Vec Operator Algebra

A is a $m \times n$ matrix and B is a $n \times p$ matrix then:

$$vec(AB) = (B^T \otimes I_m)vec(A) = (B^T \otimes A)vec(I_n) = (I_p \otimes A)vec(B)$$

(34)
Table 5: Comparison results for mining omitted activities of cascades in random and non-random deletion.

| Dataset | Metrics       | Random Missing | Non-Random Missing |
|---------|---------------|----------------|-------------------|
|         |               | DiffStru | DeepDiff | Reg-1 | Reg-2 | DiffStru | DeepDiff | Reg-1 | Reg-2 |
| LFR100  | RMSE          | 0.97     | 0.94     | 0.98  | 0.87  | 0.98     | 0.95     | 0.83  | 0.89  |
|         | MAP@10        | 70.90    | ×        | ×     | ×     | 70.20    | ×        | ×     | ×     |
|         | MAP@50        | 10.20    | ×        | ×     | ×     | 12.20    | ×        | ×     | ×     |
|         | MAP@100       | 10.60    | ×        | ×     | ×     | 13.50    | ×        | ×     | ×     |
| LFR400  | RMSE          | 0.48     | 0.26     | 0.24  | 0.49  | 0.03     | 0.26     | 0.23  |       |
|         | MAP@10        | 1.11     | ×        | ×     | ×     | 1.30     | ×        | ×     | ×     |
|         | MAP@50        | 1.60     | ×        | ×     | ×     | 1.80     | ×        | ×     | ×     |
|         | MAP@100       | 1.90     | ×        | ×     | ×     | 2.00     | ×        | ×     | ×     |
| Twitter | RMSE          | 0.18     | 0.16     | 0.16  | 0.01  | 0.07     | 0.06     | 0.08  |       |
|         | MAP@10        | 12.70    | ×        | ×     | ×     | 5.20     | ×        | ×     | ×     |
|         | MAP@50        | 13.50    | ×        | ×     | ×     | 6.80     | ×        | ×     | ×     |
|         | MAP@100       | 1.54     | ×        | ×     | ×     | 7.30     | ×        | ×     | ×     |
| Memetracker | RMSE       | 1.79     | 1.98     | 2.55  | 3.40  | 2.66     | 2.41     | 2.97  |       |
|         | MAP@10        | 8.30     | ×        | ×     | ×     | 11.20    | ×        | ×     | ×     |
|         | MAP@50        | 10.04    | ×        | ×     | ×     | 12.00    | ×        | ×     | ×     |
|         | MAP@100       | 10.70    | ×        | ×     | ×     | 12.30    | ×        | ×     | ×     |

Table 6: Link results for random deletion.

| Dataset | Methods | Acc(%) | F-measure(%) | AUC(%) | MCC | SRE |
|---------|---------|--------|-------------|--------|-----|-----|
| LFR100  | DiffStru| 90.34  | 57.68       | 91.64  | 0.54 | 0.98|
|         | FPMF    | 89.17  | 53.63       | 91.52  | 0.51 | 0.86|
|         | CN      | 85.29  | 49.22       | 86.77  | 0.45 | 0.63|
|         | AA      | 86.85  | 49.89       | 87.53  | 0.45 | 0.71|
|         | RA      | 89.98  | 50.11       | 87.87  | 0.46 | 0.71|
| LFR400  | DiffStru| 93.58  | 58.94       | 92.41  | 0.59 | 0.85|
|         | FPMF    | 93.33  | 55.44       | 91.86  | 0.54 | 0.82|
|         | CN      | 93.08  | 58.29       | 92.14  | 0.59 | 0.79|
|         | AA      | 95.10  | 58.39       | 92.16  | 0.59 | 0.79|
|         | RA      | 93.09  | 58.40       | 92.15  | 0.59 | 0.79|
| Twitter | DiffStru| 99.50  | 96.40       | 98.76  | 0.96 | 3.34|
|         | FPMF    | 99.29  | 95.00       | 96.52  | 0.94 | 0.19|
|         | CN      | 99.30  | 96.43       | 97.30  | 0.96 | 4.49|
|         | AA      | 99.30  | 96.43       | 97.30  | 0.96 | 4.49|
|         | RA      | 99.21  | 94.48       | 97.20  | 0.94 | 2.16|
| Memetracker | DiffStru| 97.83  | 79.42       | 97.74  | 0.78 | 2.43|
|         | FPMF    | 94.04  | 51.21       | 91.08  | 0.48 | 0.95|
|         | CN      | 91.00  | 54.03       | 93.33  | 0.51 | 0.92|
|         | AA      | 90.99  | 54.50       | 93.52  | 0.52 | 0.90|
|         | RA      | 91.36  | 54.72       | 93.95  | 0.52 | 0.91|

Table 7: Link results for non-random deletion.

| Dataset | Methods | Acc(%) | F-measure(%) | AUC(%) | MCC | SRE |
|---------|---------|--------|-------------|--------|-----|-----|
| LFR100  | DiffStru| 93.65  | 58.85       | 92.15  | 0.59 | 0.86|
|         | FPMF    | 93.35  | 57.99       | 92.08  | 0.57 | 0.86|
|         | CN      | 93.08  | 58.38       | 91.91  | 0.59 | 0.79|
|         | AA      | 93.09  | 58.38       | 91.94  | 0.59 | 0.79|
|         | RA      | 93.09  | 58.40       | 91.94  | 0.59 | 0.79|
| LFR400  | DiffStru| 99.82  | 98.75       | 99.84  | 0.99 | 40.45|
|         | FPMF    | 99.63  | 97.48       | 98.79  | 0.97 | 19.93|
|         | CN      | 99.84  | 98.90       | 99.48  | 0.98 | 45.93|
|         | AA      | 99.84  | 98.90       | 99.48  | 0.98 | 45.93|
|         | RA      | 99.39  | 96.78       | 99.39  | 0.96 | 13.84|
| Twitter | DiffStru| 70.25  | 80.99       | 98.33  | 0.80 | 2.65|
|         | FPMF    | 93.43  | 51.85       | 92.18  | 0.48 | 0.97|
|         | CN      | 93.27  | 55.19       | 94.58  | 0.53 | 0.88|
|         | AA      | 93.17  | 55.75       | 94.51  | 0.53 | 0.88|
|         | RA      | 94.08  | 58.03       | 95.46  | 0.53 | 1.00|

.2 Sampling \(Y\) (Equation 10)

Using the property (4.1) equation (8) yields to (35):
\[
P(\text{vec}(Y)|-)
= \exp[-\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \Pi_{ij}(C_{ij} - X^T Y_{ij})^2] \times \exp[-\frac{1}{2} \sum_{d=1}^{D} (Y_{ad} W_{vd})^2] \times \exp[-\frac{1}{2} \text{tr}(\theta C^{-1} M) (\theta C^{-1} - \theta C^{-1} Y Y^T) \times \exp[-\frac{1}{2} \text{tr}(\theta C^{-1} M) (\theta C^{-1} - \theta C^{-1} Y Y^T)]]
\]

\((35)\)
To convert $\exp[-\frac{1}{2}\text{tr}(YW^{-1}Y^T)]$ to a vec format we can multiply $I_D$ in trace.

$$
\exp[-\frac{1}{2}\text{tr}(YW^{-1}Y^T)] = \exp[-\frac{1}{2}\text{tr}(I_D(YW^{-1}Y^T))] 
$$

$$
\implies N_{MD}(\text{vec}(Y^T)^T\text{vec}(0_{MD}), (I_D \otimes W_Y)) 
$$

Using property (4.2) and Equation (36), Equation (35) can be written as:

$$
P(\text{vec}(Y)|-) = [N_{NM}(\text{vec}(\sqrt{\Pi} \circ C)|\text{vec}(\sqrt{\Pi} \circ (X^TY)), (\sigma^2_{\Pi} I_M \otimes I_N))] \times [N_{DM}(\text{vec}(Y)|\text{vec}(0), (I_D \otimes W_X))] 
$$

Then:

$$
P(\text{vec}(Y)|-) = \left[\text{vec}(\sqrt{\Pi} \circ C) - \text{vec}(\sqrt{\Pi} \circ (X^TY))\right]^{T}(\sigma^2_{\Pi} I_M \otimes I_N)^{-1} 

\times \left[\text{vec}(\sqrt{\Pi} \circ C) - \text{vec}(\sqrt{\Pi} \circ (X^TY))\right] 

+ \text{vec}(Y)^{T}(I_D \otimes W_Y)^{-1}\text{vec}(Y)) 

+ \left[\text{vec}(\sqrt{\Pi}) \circ (I_M \otimes X^T)\text{vec}(Y))

+ \left[\text{vec}(Y)^{T}(I_D \otimes W_Y)^{-1}\text{vec}(Y)) 

To have form of Equation (9)

$$
P(\text{vec}(Y)|-) = \left[\text{vec}(Y) - \mu_Y\right]^{T}\sum_{Y}^{-1}\text{vec}(Y) - \mu_Y 
$$

So for finding $\sum_{Y}$ we should look forward for term $(\text{vec}(Y))^{T}\sum_{Y}^{-1}\text{vec}(Y)$ which is:

$$
\left[\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(Y))^{T}(I_M \otimes X)](\sigma^2_{\Pi} I_M \otimes I_N) 

\times \left[\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(Y))^{T}(I_M \otimes X)^{T}\text{vec}(Y))\right] 

+ \left[\text{vec}(\sqrt{\Pi}) \circ (I_M \otimes X^T)\text{vec}(Y)) 

+ \left[\text{vec}(Y)^{T}(I_D \otimes W_Y)^{-1}\text{vec}(Y)) 

Therefore,

$$
\sum_{Y} = \left[[I_M \otimes X][\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(\sqrt{\Pi}))^{T}(I_M \otimes X)^{T}\text{vec}(C)]) 

= \left[[I_M \otimes X]\circ (\sigma^2_{\Pi} I_M \otimes I_N) 

\times \left[\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(Y))^{T}(I_M \otimes X)^{T}\text{vec}(C)]) 

+ \left[\text{vec}(Y)^{T}(I_D \otimes W_Y)^{-1}\text{vec}(Y)) 

In the same way, $\mu_Y$ can be obtained from $(\text{vec}(Y))^{T}\sum_{Y}^{-1}\mu_Y$:

$$
\mu_Y = \sum_{Y}[I_M \otimes X]\circ (\sigma^2_{\Pi} I_M \otimes I_N) 

\times \left[\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(Y))^{T}(I_M \otimes X)^{T}\text{vec}(C)]) 

+ \left[\text{vec}(Y)^{T}(I_D \otimes W_Y)^{-1}\text{vec}(Y)) 

3 Sampling X (Equation 25)

Following the property (4.1) Equation (21) yields to (43):

$$
P(\text{vec}(X)|-) = [N_{NM}(\text{vec}(\sqrt{\Pi} \circ C)|\text{vec}(\sqrt{\Pi} \circ (X^TY)), (\sigma^2_{\Pi} I_M \otimes I_N))] \times [N_{NN}(\text{vec}(R)|\text{vec}(X^T U), (\sigma^2_{\Pi} I_M \otimes I_N))] 

\times [N_{DM}(\text{vec}(X)|\text{vec}(0), (I_D \otimes W_X))] 
$$

For writing as Equation (22), we have:

$$
P(\text{vec}(X)|-) = \left[\text{vec}(X) - \mu_X\right]^{T}\sum_{X}^{-1}\text{vec}(X) - \mu_X 
$$

So for finding $\sum_{X}$ term $(\text{vec}(X))^{T}\sum_{X}^{-1}\text{vec}(X)$ is:

$$
\left[\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(X))^{T}(Y \otimes I_N)(\sigma^2_{\Pi} I_N \otimes I_N) 

\times \left[\text{vec}(\sqrt{\Pi}) \circ (Y^T \otimes I_N)\text{vec}(X^T)]) 

+ \left[\text{vec}(X)^{T}(U \otimes I_N)\text{vec}(X^T)]) 

+ \left[\text{vec}(X)^{T}(U \otimes I_N)\text{vec}(X^T)]) 

Finally,

$$
\sum_{X} = \left[[Y \otimes I_N][\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(\sqrt{\Pi}))^{T}(Y \otimes I_N)(\sigma^2_{\Pi} I_N \otimes I_N) 

\times \left[\text{vec}(\sqrt{\Pi}) \circ (Y^T \otimes I_N)\text{vec}(X^T)]) 

+ \left[\text{vec}(X)^{T}(U \otimes I_N)(\sigma^2_{\Pi} I_N \otimes I_N) 

+ \left[\text{vec}(X)^{T}(U \otimes I_N)(\sigma^2_{\Pi} I_N \otimes I_N) 

+ \left[\text{vec}(X)^{T}(U \otimes I_N)(\sigma^2_{\Pi} I_N \otimes I_N) 

In the same way, $\mu_X$ can be obtained from $(\text{vec}(X))^{T}\sum_{X}^{-1}\mu_X$:

$$
\mu_X = \sum_{X}[Y \otimes I_N]\circ (\sigma^2_{\Pi} I_M \otimes I_N) 

\times \left[\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(Y))^{T}(Y \otimes I_N)(\sigma^2_{\Pi} I_M \otimes I_N) 

+ \left[\text{vec}(Y)^{T}(Y \otimes I_N)\text{vec}(Y) 

+ \left[\text{vec}(Y)^{T}(Y \otimes I_N)\text{vec}(Y) 

Then from $(\text{vec}(X))^{T}\sum_{X}^{-1}\mu_X T, \mu_X T$ is:

$$
\left[\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(X))^{T}(Y \otimes I_N)(\sigma^2_{\Pi} I_N \otimes I_N) 

\times \left[\text{vec}(\sqrt{\Pi}) \circ (Y^T \otimes I_N)\text{vec}(X^T)]) 

+ \left[\text{vec}(X)^{T}(U \otimes I_N)(\sigma^2_{\Pi} I_N \otimes I_N) 

+ \left[\text{vec}(X)^{T}(U \otimes I_N)(\sigma^2_{\Pi} I_N \otimes I_N) 

In the same way, $\mu_Y$ can be obtained from $(\text{vec}(Y))^{T}\sum_{Y}^{-1}\mu_Y$:

$$
\mu_Y = \sum_{Y}[I_M \otimes X]\circ (\sigma^2_{\Pi} I_M \otimes I_N) 

\times \left[\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(Y))^{T}(I_M \otimes X)(\sigma^2_{\Pi} I_M \otimes I_N) 

+ \left[\text{vec}(Y)^{T}(I_M \otimes X)^{T}\text{vec}(C)]) 

+ \left[\text{vec}(Y)^{T}(I_M \otimes X)^{T}\text{vec}(C)]) 

Finally,

$$
\sum_{X} = \left[[Y \otimes I_N][\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(\sqrt{\Pi}))^{T}(Y \otimes I_N)(\sigma^2_{\Pi} I_N \otimes I_N) 

+ \left[\text{vec}(Y)^{T}(Y \otimes I_N)\text{vec}(Y) 

+ \left[\text{vec}(Y)^{T}(Y \otimes I_N)\text{vec}(Y) 

In the same way, $\mu_Y$ can be obtained from $(\text{vec}(Y))^{T}\sum_{Y}^{-1}\mu_Y$:

$$
\mu_Y = \sum_{Y}[I_M \otimes X]\circ (\sigma^2_{\Pi} I_M \otimes I_N) 

\times \left[\text{vec}(\sqrt{\Pi}) \circ (\text{vec}(Y))^{T}(I_M \otimes X)(\sigma^2_{\Pi} I_M \otimes I_N) 

+ \left[\text{vec}(Y)^{T}(I_M \otimes X)^{T}\text{vec}(C)]) 

+ \left[\text{vec}(Y)^{T}(I_M \otimes X)^{T}\text{vec}(C)]) 

Finally,
