Network Diffusions via Neural Mean-Field Dynamics

Shushan He
Mathematics & Statistics
Georgia State University
Atlanta, Georgia, USA
she4@gsu.edu

Hongyuan Zha
School of Data Science
Shenzhen Research Institute of
Big Data, CUHK, Shenzhen, China
zhahy@cuhk.edu.cn

Xiaojing Ye
Mathematics & Statistics
Georgia State University
Atlanta, Georgia, USA
xye@gsu.edu

Abstract

We propose a novel learning framework based on neural mean-field dynamics for *simultaneous* inference and estimation problems of diffusions on networks. Our new framework is derived from the Mori-Zwanzig formalism to obtain an exact evolution of the node infection probabilities, which renders a delay differential equation with memory integral approximated by learnable time convolution operators, resulting in a highly structured and interpretable RNN. Directly using cascade data, our framework can *jointly* learn the structure of the diffusion network and the evolution of infection probabilities, which are cornerstone to important downstream applications such as influence maximization. Connections between parameter learning and optimal control are also established. Empirical study shows that our approach is versatile and robust to variations of the underlying diffusion network models, and significantly outperforms existing approaches in accuracy and efficiency on both synthetic and real-world data.

1 Introduction

Continuous-time information diffusion on heterogeneous networks is a prevalent phenomenon [4, 39, 43]. News spreading on social media [13, 15, 49], viral marketing [23, 25, 52], computer malware propagation, and epidemics of contagious diseases [3, 36, 43, 47] are all examples of diffusion on networks, among many others. For instance, a piece of information (such as a tweet) can be retweeted by users (nodes) with followee-follower relationships (edge) on the Twitter network. We call a user *infected* if she retweets, and her followers see her retweet and can also become infected if they retweet in turn, and so on. Such information diffusion mimics the epidemic spread where an infectious virus can spread to individuals (human, animal, or plant) and then to many others upon their close contact.

In this paper, we are mainly concerned with the estimation of individual node infection probabilities as well as inference of the underlying diffusion network structures directly using cascade data of historical diffusion events on the network. For infection probability estimation, our goal is to compute the evolution of the probability of each node being infected during a diffusion initiated from a set of source nodes. For network structure inference, we aim at learning the edges as well as the strength of interactions (through the edges) between the nodes on the diffusion network. Not surprisingly, both problems are very challenging due to the extremely large scale of modern networks, the heterogeneous inter-dependencies among the nodes, and the randomness exhibited in cascade data. Most existing works focus on one problem only, e.g., either to solely infer the network structure from cascade data, or to estimate influence without providing insights into the underlying network structure.

We propose a novel learning framework, called neural mean-field (NMF) dynamics, to *simultaneously* tackle both of the estimation and inference problems mentioned above. Specifically: (i) We develop a neural mean-field dynamics framework to model the evolution of diffusion on a network. Our new framework is derived from the Mori-Zwanzig formalism to obtain an *exact* time evolution of the node infection probability with dimension linear in the network size; (ii) We show that the memory term of

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

the Mori-Zwanzig equation can be approximated by a trainable convolution network, which renders the dynamical system into a delay differential equation. We also show that the time discretization of such system reduces to a recurrent neural network. The approximate system is highly *interpretable*, and in particular, the training accepts sample cascades as input, and returns both individual probability estimates (and hence the influence function) as well as structure information of the diffusion network as outputs; (iii) We show that the parameters learning in NMF can be reduced to an optimal control problem with the parameters as time invariant control inputs, maximizing the *total Hamiltonian* of the system; and (iv) Our empirical analysis shows that our approach is robust to the variation of the unknown underlying diffusion models, and it also significantly outperforms existing approaches for both synthetic and real-world diffusion networks.

The remainder of this paper is organized as follows. In Section 2, we introduce the diffusion network models and related background information, including the influence predication and structure inference problems. In Section 3, we develop the proposed framework of neural mean-field dynamics for inference and prediction on diffusion networks, as well as an optimal control formulation for parameter learning. We demonstrate the performance of the proposed method on influence estimation and maximization on a variety of synthetic and real-world networks in Section 4. A discussion of the related work is given in Section 5. Section 6 concludes the paper.

2 Preliminaries on Diffusion Networks

Throughout this paper, we use boldfaced lower (upper) letter to denote vector (matrix) or vector-valued (matrix-valued) function, and $(\cdot)_k$ (or $(\cdot)_{ij}$) for its kth component (or (i,j)-th entry). All vectors are column vectors unless otherwise noted. We follow the Matlab syntax and use $[\boldsymbol{x};\boldsymbol{y}]$ to denote the vector that stacks $\boldsymbol{x}$ and $\boldsymbol{y}$ vertically. We denote inner product by $\boldsymbol{x}\cdot\boldsymbol{y}$ and component-wise multiplication by $\boldsymbol{x}\odot\boldsymbol{y}$. Time is denoted by t in either continuous ($t\in[0,T]$) or discrete case ($t=0,1,\ldots,T$) for some time horizon $T\in\mathbb{R}_+$ ($\mathbb{N}$ in discrete case). Derivative $'$ is with respect to t, and gradient $\nabla_{\boldsymbol{x}}$ is with respect to $\boldsymbol{x}$. Probability is denoted by $\Pr(\cdot)$, and expectation with respect to X (or p_X) is denoted by $\mathbb{E}_X[\,\cdot\,]$.

Diffusion network models Consider a diffusion network model, which consists of a network (directed graph) $\mathcal{G}=(\mathcal{V},\mathcal{E})$ with node set $\mathcal{V}=[n]:=\{1,\ldots,n\}$ and edge set $\mathcal{E}\subset\mathcal{V}\times\mathcal{V}$, and a *diffusion model* that describes the distribution $p(t;\alpha_{ij})$ of the time t node i takes to infect a healthy neighbor $j\in\{j':(i,j')\in\mathcal{E}\}$ for every $(i,j)\in\mathcal{E}$. Then, given a source (seed) set $\mathcal{S}$ of nodes that are infected at time 0, they will infect their healthy neighbors with infection time following p, and the infected neighbors will then infect their healthy neighbors, and so on, such that the infection initiated from $\mathcal{S}$ at time 0 propagates to other nodes of the network.

Typical diffusion network models are assumed to be *progressive* where infected node cannot recover and the infections on different edges are independent. For example, the standard diffusion model with exponential distribution $p(t;\alpha)=\alpha e^{-\alpha t}$ is mostly widely used; other distributions can also be considered, as is done in this paper. For simplicity, we focus on uni-parameter distributions or distributions with multiple parameters but only one can vary across different edges with the consequence that the parameter $\alpha_{ij}\geq 0$ indicates the *strength* of impact node i has on node j.

Cascade data Observation data $\mathcal{D}$ of a diffusion network are often in the form of *sample cascades* $\mathcal{D}:=\{\mathcal{C}_k=(\mathcal{S}_k,\boldsymbol{\tau}_k)\in 2^{\mathcal{V}}\times\mathbb{R}_+^n : k\in[K]\}$, where the kth cascade $\mathcal{C}_k$ records its source set $\mathcal{S}_k\subset\mathcal{V}$ and the time $(\boldsymbol{\tau}_k)_i\geq 0$ which indicates when node i was infected (if i was not infected during $\mathcal{C}_k$ then $(\boldsymbol{\tau}_k)_i=\infty$). We also equate $\mathcal{C}_k$ with $\{\hat{\boldsymbol{x}}^{(k)}(t)\in\{0,1\}^n : i\in[n], t\geq 0\}$ such that $(\hat{\boldsymbol{x}}^{(k)}(t))_i=1$ if the node i is in the infected status at time t and 0 otherwise. For example, $\hat{\boldsymbol{x}}^{(k)}(0)=\boldsymbol{\chi}_{\mathcal{S}_k}$ where $(\boldsymbol{\chi}_{\mathcal{S}_k})_i=1$ if $i\in\mathcal{S}_k$ and 0 otherwise. Such cascade data are collected from historical events for training purposes.

Influence prediction and inference of diffusion network Given the network $\mathcal{G}=(\mathcal{V},\mathcal{E})$, as well as the diffusion model and $\boldsymbol{A}$, where $(\boldsymbol{A})_{ji}=\alpha_{ij}$ is the parameter of $p(t;\alpha_{ij})$ for edge (i,j), the *inference prediction* (or *influence estimation*) is to compute

$$\boldsymbol{x}(t;\boldsymbol{\chi}_{\mathcal{S}})=[x_1(t;\boldsymbol{\chi}_{\mathcal{S}}),\ldots,x_n(t;\boldsymbol{\chi}_{\mathcal{S}})]^\top\in[0,1]^n \tag{1}$$

for all time $t \geq 0$ and any source set $\mathcal{S} \subset \mathcal{V}$. In (1), $x_i(t; \boldsymbol{\chi}_\mathcal{S})$ is the probability of node i being infected at time t given a source set $\mathcal{S}$ (not necessarily observed as a source set in $\mathcal{D}$). Note that we use $\boldsymbol{\chi}_\mathcal{S}$ and $\mathcal{S}$ interchangeably hereafter. The probability $\boldsymbol{x}(t; \boldsymbol{\chi}_\mathcal{S})$ can also be used to compute the *influence function* $\sigma(t; \mathcal{S}) := \mathbf{1}_n^\top \boldsymbol{x}(t; \boldsymbol{\chi}_\mathcal{S})$, the expected number of infected nodes at time t. Note that an analytic solution of (1) is intractable due to the exponentially large state space of the complete dynamical system of the diffusion problem [19, 48].

On the other hand, network *inference* refers to learning the network connectivity $\mathcal{E}$ and $\boldsymbol{A}$ given cascade data $\mathcal{D}$. The matrix $\boldsymbol{A}$ is the distribution parameters if the diffusion model p is given, or it simply qualitatively measures the strength of impact node i on j if no specific p is known.

Influence prediction may also require network inference when only cascade data $\mathcal{D}$ are available, resulting in a *two-stage* approach: a network inference is performed first to learn the network structure $\mathcal{E}$ and the diffusion model parameters $\boldsymbol{A}$, and then an influence estimation is used to compute the influence for the source set $\mathcal{S}$. However, approximation errors and biases in the two stages will certainly accumulate. Alternatively, one can use a *one-stage* approach to directly estimate $\boldsymbol{x}(t; \boldsymbol{\chi}_\mathcal{S})$ of any $\mathcal{S}$ from the cascade data $\mathcal{D}$, which is more versatile and less prone to diffusion model misspecification. Our method is a such kind of one-stage method. Additionally, it allows knowledge of $\mathcal{E}$ and/or $\boldsymbol{A}$, if available, to be integrated for further performance improvement.

Influence maximization Given cascade data $\mathcal{D}$, *influence maximization* is to find the source set $\mathcal{S}$ that generates the maximal influence $\sigma(t; \mathcal{S})$ at t among all subsets of size n_0, where $t > 0$ and $1 \leq n_0 < n$ are prescribed. Namely, influence maximization can be formulated as

$$\max_{\mathcal{S}} \ \sigma(t; \mathcal{S}), \quad \text{s.t.} \quad \mathcal{S} \subset \mathcal{V}, \quad |\mathcal{S}| \leq n_0. \tag{2}$$

There are two main ingredients of an influence maximization method for solving (2): an influence prediction subroutine that evaluates the influence $\sigma(t; \mathcal{S})$ for any given source set $\mathcal{S}$, and an (approximate) combinatorial optimization solver to find the optimal set $\mathcal{S}$ of (2) that repeatedly calls the subroutine. The combinatorial optimization problem is NP-hard and is often approximately solved by greedy algorithms with guaranteed sub-optimality when $\sigma(t; \mathcal{S})$ is submodular in $\mathcal{S}$. In our experiment, we show that a standard greedy approach equipped with our proposed influence estimation method outperforms other state-of-the-art influence maximization algorithms.

3 Neural Mean-Field Dynamics

Modelling diffusion by stochastic jump processes We begin with the jump process formulation of network diffusion. Given a source set $\boldsymbol{\chi}_\mathcal{S}$, let $X_i(t; \boldsymbol{\chi}_\mathcal{S})$ denote the infection status of the node i at time t. Namely, $X_i(t) = 1$ if node i is infected by time t, and 0 otherwise. Then $\{X_i(t) : i \in [n]\}$ are a set of n coupled jump processes, such that $X_i(t; \boldsymbol{\chi}_\mathcal{S})$ jumps from 0 to 1 when the node i is infected by any of its infected neighbors at t. Let $\lambda_i^*(t)$ be the conditional intensity of $X_i(t; \boldsymbol{\chi}_\mathcal{S})$ given the history $\mathcal{H}(t) = \{X_i(s; \boldsymbol{\chi}_\mathcal{S}) : s \leq t,\, i \in [n]\}$, i.e.,

$$\lambda_i^*(t) := \lim_{\tau \to 0^+} \frac{\mathbb{E}[X_i(t + \tau; \boldsymbol{\chi}_\mathcal{S}) - X_i(t; \boldsymbol{\chi}_\mathcal{S}) | \mathcal{H}(t)]}{\tau}. \tag{3}$$

Note that the numerator of (3) is also the conditional probability $\Pr(X_i(t + \tau) = 1, X_i(t) = 0 | \mathcal{H}(t))$ for any $\tau > 0$. In influence prediction, our goal is to compute the probability $\boldsymbol{x}(t; \boldsymbol{\chi}_\mathcal{S}) = [x_i(t; \boldsymbol{\chi}_\mathcal{S})]$ in (1), which is the expectation of $X_i(t; \boldsymbol{\chi}_\mathcal{S})$ conditioning on $\mathcal{H}(t)$:

$$x_i(t; \boldsymbol{\chi}_\mathcal{S}) = \mathbb{E}_{\mathcal{H}(t)}[X_i(t; \boldsymbol{\chi}_\mathcal{S}) | \mathcal{H}(t)]. \tag{4}$$

To this end, we adopt the following notations (for notation simplicity we temporarily drop $\boldsymbol{\chi}_\mathcal{S}$ in this subsection as the source set $\mathcal{S}$ is arbitrary but fixed):

$$x_I(t) = \mathbb{E}_{\mathcal{H}(t)} \left[\textstyle\prod_{i \in I} X_i(t; \boldsymbol{\chi}_\mathcal{S}) \big| \mathcal{H}(t) \right], \quad y_I(t) = \textstyle\prod_{i \in I} x_i(t), \quad e_I(t) = x_I(t) - y_I(t) \tag{5}$$

for any $I \subset [n]$ and $|I| \geq 2$. Then we can derive the evolution of $\boldsymbol{z} := [\boldsymbol{x}; \boldsymbol{e}]$. Here $\boldsymbol{x}(t) \in [0, 1]^n$ is the *resolved* variable whose value is of interests and samples can be directly observed from the cascade data $\mathcal{D}$, and $\boldsymbol{e}(t) = [\cdots, e_I(t), \cdots] \in \mathbb{R}^{N-n}$ where $N = 2^n - 1$ is the *unresolved* variable that captures all the second and higher order moments. The complete evolution equation of $\boldsymbol{z}$ is given in the following theorem, where the proof is provided in Appendix B.1.

Theorem 1. *The evolution of $\boldsymbol{z}(t) = [\boldsymbol{x}(t); \boldsymbol{e}(t)]$ follows the nonlinear differential equation:*

$$\boldsymbol{z}' = \bar{\boldsymbol{f}}(\boldsymbol{z}), \quad \textit{where} \quad \bar{\boldsymbol{f}}(\boldsymbol{z}) = \bar{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{e}) = \left[\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{A}) - (\boldsymbol{A} \odot \boldsymbol{E})\mathbf{1};\ \cdots, f_I(\boldsymbol{x}, \boldsymbol{e}); \cdots\right], \tag{6}$$

with initial value $\boldsymbol{z}_0 = [\boldsymbol{\chi}_{\mathcal{S}}; \mathbf{0}] \in \mathbb{R}^N$, $\boldsymbol{E} = [e_{ij}] \in \mathbb{R}^{n \times n}$, and

$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{A}) = \boldsymbol{A}\boldsymbol{x} - \operatorname{diag}(\boldsymbol{x})\boldsymbol{A}\boldsymbol{x}, \tag{7}$$

$$f_I(\boldsymbol{x}, \boldsymbol{e}) = \sum_{i \in I} \sum_{j \notin I} \alpha_{ji}(y_I - y_{I \cup \{j\}} + e_I - e_{I \cup \{j\}}) - \sum_{i \in I} y_{I \setminus \{i\}} \sum_{j \neq i} \alpha_{ji}(x_j - y_{ij} - e_{ij}). \tag{8}$$

The evolution (6) holds true exactly for the standard diffusion model with exponential distribution, but also approximates well for other distributions p, as shown in the empirical study below. In either case, the dimension N of $\boldsymbol{z}$ grows exponentially fast in network size n and hence renders the computation infeasible in practice. To overcome this issue, we employ the Mori-Zwanzig formalism [7] to derive a reduced-order model of $\boldsymbol{x}$ with dimensionality n only.

Mori-Zwanzig memory closure We employ the Mori-Zwanzig (MZ) formalism [7] that allows to introduce a generalized Langevin equation (GLE) of the $\boldsymbol{x}$ part of the dynamics (6). The GLE of $\boldsymbol{x}$ is derived from the original equation (6) describing the evolution of $\boldsymbol{z} = [\boldsymbol{x}; \boldsymbol{e}]$, while maintaining the effect of the unresolved part $\boldsymbol{e}$. This is particularly useful in our case, as we only need $\boldsymbol{x}$ for infection probability estimation and influence prediction.

Define the Liouville operator $\mathcal{L}$ such that $\mathcal{L}[g](\boldsymbol{z}) := \bar{\boldsymbol{f}}(\boldsymbol{z}) \cdot \nabla_{\boldsymbol{z}} g(\boldsymbol{z})$ for any real-valued function g of $\boldsymbol{z}$. Let $e^{t\mathcal{L}}$ be the Koopman operator associated with $\mathcal{L}$ such that $e^{t\mathcal{L}} g(\boldsymbol{z}(0)) = g(\boldsymbol{z}(s))$ where $\boldsymbol{z}(t)$ solves (6). Then $\mathcal{L}$ is known to satisfy the semi-group property for all g, i.e., $e^{t\mathcal{L}} g(\boldsymbol{z}) = g(e^{t\mathcal{L}}\boldsymbol{z})$. Now consider the projection operator $\mathcal{P}$ as the truncation such that $(\mathcal{P}g)(\boldsymbol{z}) = (\mathcal{P}g)([\boldsymbol{x}; \boldsymbol{e}]) = g([\boldsymbol{x}; 0])$ for any $\boldsymbol{z} = [\boldsymbol{x}; \boldsymbol{e}]$, and its orthogonal complement as $\mathcal{Q} = I - \mathcal{P}$ where I is the identity operator. The following theorem describes the *exact* evolution of $\boldsymbol{x}(t)$, and the proof is given in Appendix B.2.

Theorem 2. *The evolution of $\boldsymbol{x}$ specified in* (6) *can also be described by the following GLE:*

$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{A}) + \int_0^t \boldsymbol{k}(t - s, \boldsymbol{x}(s))\, \mathrm{d}s, \tag{9}$$

where $\boldsymbol{f}$ is given in (7)*, and $\boldsymbol{k}(t, \boldsymbol{x}) := \mathcal{P}\mathcal{L}e^{t\mathcal{Q}\mathcal{L}}\mathcal{Q}\mathcal{L}\boldsymbol{x}$.*

Note that, (9) is *not* an approximation—it is an *exact* representation of the $\boldsymbol{x}$ part of the original problem (6). The equation (9) can be interpreted as a *mean-field* equation, where the two terms on the right hand side are called the *streaming term* (corresponding to the mean-field dynamics) and *memory term*, respectively. The streaming term provides the *main drift* of the evolution, and the memory term in the convolution form is for vital *adjustment*. This inspires us to approximate the memory term as a time convolution on $\boldsymbol{x}$, which naturally yields a delay differential equation and further reduces to a structured recurrent neural network (RNN) after discretization, as shown in the next subsection.

Delay differential equation and RNN To compute the evolution (9) of $\boldsymbol{x}$, we consider an approximation of the Mori-Zwanzig memory term by a neural net $\boldsymbol{\varepsilon}$ with time convolution of $\boldsymbol{x}$ as follows,

$$\int_0^t \boldsymbol{k}(t - s, \boldsymbol{x}(s))\, \mathrm{d}s \approx \boldsymbol{\varepsilon}(\boldsymbol{x}(t), \boldsymbol{h}(t); \boldsymbol{\eta}) \quad \text{where} \quad \boldsymbol{h}(t) = \int_0^t \boldsymbol{K}(t - s; \boldsymbol{w})\boldsymbol{x}(s)\, \mathrm{d}s. \tag{10}$$

In (10), $\boldsymbol{K}(\cdot; \boldsymbol{w})$ is a convolutional operator with parameter $\boldsymbol{w}$, and $\boldsymbol{\varepsilon}(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\eta})$ is a deep neural net with $(\boldsymbol{x}, \boldsymbol{h})$ as input and $\boldsymbol{\eta}$ as parameter. Both $\boldsymbol{w}$ and $\boldsymbol{\eta}$ are to be trained by the cascade data $\mathcal{D}$. Hence, (9) reduces to a *delay differential equation* which involves a time integral $\boldsymbol{h}(t)$ of past $\boldsymbol{x}$:

$$\boldsymbol{x}' = \tilde{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta}) := \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{A}) + \boldsymbol{\varepsilon}(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\eta}). \tag{11}$$

The initial condition of (11) with source set $\mathcal{S}$ is given by

$$\boldsymbol{x}(0) = \boldsymbol{\chi}_{\mathcal{S}}, \quad \boldsymbol{h}(0) = \mathbf{0}, \quad \text{and} \quad \boldsymbol{x}(t) = \boldsymbol{h}(t) = \mathbf{0}, \quad \forall\, t < 0. \tag{12}$$

We call the system (11) with initial (12) the *neural mean-field* (NMF) dynamics.

The delay differential equation (11) is equivalent to a coupled system of $(\boldsymbol{x}, \boldsymbol{h})$. In addition, we show that the discretization of this system reduces to a structured recurrent neural network if $\boldsymbol{K}(t; \boldsymbol{w})$ is a (linear combination of) matrix convolutions in the following theorem.

Theorem 3. *The delay differential equation* (11) *is equivalent to the following coupled system:*

$$\boldsymbol{x}' = \tilde{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{A}, \boldsymbol{\eta}) = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{A}) + \boldsymbol{\varepsilon}(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\eta}) \tag{13a}$$

$$\boldsymbol{h}' = \textstyle\int_0^t \boldsymbol{K}(t-s; \boldsymbol{w}) \tilde{\boldsymbol{f}}(\boldsymbol{x}(s), \boldsymbol{h}(s); \boldsymbol{A}, \boldsymbol{\eta}) \, \mathrm{d}s \tag{13b}$$

with initial condition (12). *In particular, if* $\boldsymbol{K}(t; \boldsymbol{w}) = \sum_{l=1}^L \boldsymbol{B}_l e^{-\boldsymbol{C}_l t}$ *for some* $L \in \mathbb{N}$ *with* $\boldsymbol{w} = \{(\boldsymbol{B}_l, \boldsymbol{C}_l)_l : \boldsymbol{B}_l \boldsymbol{C}_l = \boldsymbol{C}_l \boldsymbol{B}_l, \, \forall \, l \in [L]\}$, *then* (13) *can be solved by a non-delay system of* $(\boldsymbol{x}, \boldsymbol{h})$ *with* (13a) *and* $\boldsymbol{h}' = \sum_{l=1}^L (\boldsymbol{B}_l \boldsymbol{x} - \boldsymbol{C}_l \boldsymbol{h})$. *The discretization of such system (with step size normalized to 1) reduces to an RNN with hidden layers* $(\boldsymbol{x}_t, \boldsymbol{h}_t)$ *for* $t = 0, 1, \ldots, T-1$*:*

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{f}(\boldsymbol{x}_t; \boldsymbol{A}) + \boldsymbol{\varepsilon}(\boldsymbol{x}_t, \boldsymbol{h}_t; \boldsymbol{\eta}) \tag{14a}$$

$$\boldsymbol{h}_{t+1} = \boldsymbol{h}_t + \textstyle\sum_{l=1}^L (\boldsymbol{B}_l \boldsymbol{x}_{t+1} - \boldsymbol{C}_l \boldsymbol{h}_t) \tag{14b}$$

where the input is given by $\boldsymbol{x}_0 = \boldsymbol{\chi}_{\mathcal{S}}$ *and* $\boldsymbol{h}_0 = \boldsymbol{0}$.

The proof is given in Appendix B.3. The matrices $\boldsymbol{B}_l$ and $\boldsymbol{C}_l$ in (14b) correspond to the weights on $\boldsymbol{x}_{t+1}$ and $\boldsymbol{h}_t$ to form the *linear* transformation, and the neural network $\boldsymbol{\varepsilon}$ wraps up $(\boldsymbol{x}_t, \boldsymbol{h}_t)$ to approximate the *nonlinear* effect of the memory term in (10).

We here consider a more general convolution kernel $\boldsymbol{K}(\cdot; \boldsymbol{w})$ than the exponential kernel. Note that, in practice, the convolution weight $\boldsymbol{K}$ on older state $\boldsymbol{x}$ in (10) rapidly diminishes, and hence the memory kernel $\boldsymbol{K}$ can be well approximated with a truncated history of finite length $\tau > 0$, or $\tau \in \mathbb{N}$ after discretization. Hence, we substitute (14b) by

$$\boldsymbol{h}_t = \boldsymbol{K}^{\boldsymbol{w}} \boldsymbol{m}_t \quad \text{where} \quad \boldsymbol{K}^{\boldsymbol{w}} = [\boldsymbol{K}_0^{\boldsymbol{w}}, \ldots, \boldsymbol{K}_\tau^{\boldsymbol{w}}] \quad \text{and} \quad \boldsymbol{m}_t = [\boldsymbol{x}_t; \ldots; \boldsymbol{x}_{t-\tau}]. \tag{15}$$

Then we formulate the evolution of the augmented state $\boldsymbol{m}_t$ defined in (15) and follow (14a) to obtain a single evolution of $\boldsymbol{m}_t$ for $t = 0, \ldots, T-1$:

$$\boldsymbol{m}_{t+1} = \boldsymbol{g}(\boldsymbol{m}_t; \boldsymbol{\theta}), \;\; \text{where} \;\; \boldsymbol{g}(\boldsymbol{m}; \boldsymbol{\theta}) := [\boldsymbol{J}_0 \boldsymbol{m} + \tilde{\boldsymbol{f}}(\boldsymbol{J}_0 \boldsymbol{m}, \boldsymbol{K}^{\boldsymbol{w}} \boldsymbol{m}; \boldsymbol{\theta}); \boldsymbol{J}_0 \boldsymbol{m}; \ldots; \boldsymbol{J}_{\tau-1} \boldsymbol{m}] \tag{16}$$

and $\boldsymbol{J}_s := [\cdots, \boldsymbol{I}, \cdots] \in \mathbb{R}^{n \times (\tau+1)n}$ has identity $\boldsymbol{I}$ as the $(s+1)$th block and $\boldsymbol{0}$ elsewhere (thus $\boldsymbol{J}_s \boldsymbol{m}_t$ extracts the $(s+1)$th block $\boldsymbol{x}_{t-s}$ of $\boldsymbol{m}_t$) for $s = 0, \ldots, \tau-1$. If (14b) is considered, a simpler augmented state $\boldsymbol{m}_t = [\boldsymbol{x}_t; \boldsymbol{h}_t]$ can be formed similarly; we omit the details here. We will use the dynamics (16) of the augmented state $\boldsymbol{m}_t$ in the training below.

An optimal control formulation of parameter learning Now we consider the training of the network parameters $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{\eta}, \boldsymbol{w})$ of (16) using cascade data $\mathcal{D}$. Given a sample cascade $\hat{\boldsymbol{x}} = (\mathcal{S}, \boldsymbol{\tau})$ from $\mathcal{D}$, we can observe its value in $\{0,1\}^n$ at each of the time points $t = 1, \ldots, T$ and obtain the corresponding infection states, i.e., $\hat{\boldsymbol{x}} = \{\hat{\boldsymbol{x}}_t \in \{0,1\}^n : t \in [T]\}$ (see Section 2). Maximizing the log-likelihood of $\hat{\boldsymbol{x}}$ for the dynamics $\boldsymbol{x}_t = \boldsymbol{x}_t(\boldsymbol{\theta}) \in [0,1]^n$ induced by $\boldsymbol{\theta}$ is equivalent to minimizing the loss function $\ell(\boldsymbol{x}, \hat{\boldsymbol{x}})$:

$$\ell(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \textstyle\sum_{t=1}^T \hat{\boldsymbol{x}}_t \cdot \log \boldsymbol{x}_t + (\boldsymbol{1} - \hat{\boldsymbol{x}}_t) \cdot \log(\boldsymbol{1} - \boldsymbol{x}_t), \tag{17}$$

where the logarithm is taken componentwisely. We can add a regularization term $r(\boldsymbol{\theta})$ to (17) to impose prior knowledge or constraint on $\boldsymbol{\theta}$. In particular, if $\mathcal{E}$ is known, we can enforce a constraint such that $\boldsymbol{A}$ must be supported on $\mathcal{E}$ only. Otherwise, we can add $\|\boldsymbol{A}\|_1$ or $\|\boldsymbol{A}\|_0$ (the l_1 or l_0 norm of the vectorized $\boldsymbol{A}$) if $\mathcal{E}$ is expected to be sparse. In general, $\boldsymbol{A}$ can be interpreted as the convolution to be learned from a graph convolution network (GCN) [26, 53]. The support and magnitude of $\boldsymbol{A}$ imply the network structure and strength of interaction between nodes, respectively. We provide more details of our numerical implementation in Section 4 and Appendix D.1.

The optimal parameter $\boldsymbol{\theta}$ can be obtained by minimizing the loss function in (17) subject to the NMF dynamics (16). This procedure can also be cast as an optimal control problem to find $\boldsymbol{\theta}$ that steers $\boldsymbol{m}_t$ to fit data $\mathcal{D}$ through the NMF in (16):

$$\min_{\boldsymbol{\theta}} \quad \mathcal{J}(\boldsymbol{\theta}) := (1/K) \cdot \textstyle\sum_{k=1}^K \ell(\boldsymbol{x}^{(k)}, \hat{\boldsymbol{x}}^{(k)}) + r(\boldsymbol{\theta}) \tag{18a}$$

$$\text{s.t.} \quad \boldsymbol{m}_{t+1}^{(k)} = \boldsymbol{g}(\boldsymbol{m}_t^{(k)}; \boldsymbol{\theta}), \quad \boldsymbol{m}_0^{(k)} = [\boldsymbol{\chi}_{\mathcal{S}_k}, \boldsymbol{0}, \ldots, \boldsymbol{0}], \quad t \in [T] - 1, \; k \in [K], \tag{18b}$$

where $\boldsymbol{x}_t^{(k)} = \boldsymbol{J}_0 \boldsymbol{m}_t^{(k)}$ for all t and k. The problem of optimal control has been well studied in both continuous and discrete cases in the past decades [2]. In particular, the discrete optimal control

with nonlinear difference equations and the associated maximum principle have been extensively exploited. Recently, an optimal control viewpoint of deep learning has been proposed [32]—the network parameters of a neural network play the role of control variable in a discretized differential equation, and the training of these parameters for the network output to minimize the loss function can be viewed as finding the optimal control to minimize the objective function at the terminal state.

The Pontryagin's Maximum Principle (PMP) provides an important optimality condition of the optimal control [2, 32]. In standard optimal control, the control variable can be chosen freely in the allowed set at any given time t, which is a key in the proof of PMP. However, the NMF dynamics derived in (13) or (14) require a time invariant control $\boldsymbol{\theta}$ throughout. This is necessary since $\boldsymbol{\theta}$ corresponds to the network parameter and needs to be shared across different layers of the RNN, either from the linear kernel case with state $[\boldsymbol{x}; \boldsymbol{h}]$ in (14) or the general case with state $\boldsymbol{m}$ in (16). Therefore, we need to modify the original PMP and the optimality condition for our NMF formulation. To this end, consider the *Hamiltonian* function

$$H(\boldsymbol{m}, \boldsymbol{p}; \boldsymbol{\theta}) = \boldsymbol{p} \cdot \boldsymbol{g}(\boldsymbol{m}; \boldsymbol{\theta}) - \tfrac{1}{T} r(\boldsymbol{\theta}), \tag{19}$$

and define the *total Hamiltonian* of the system (14) as $\sum_{t=0}^{T-1} H(\boldsymbol{m}_t, \boldsymbol{p}_{t+1}; \boldsymbol{\theta})$. Then we can show that the optimal solution $\boldsymbol{\theta}^*$ is a time invariant control satisfying a *modified* PMP as follows.

Theorem 4. *Let $\boldsymbol{x}^*$ be the optimally controlled state process by $\boldsymbol{\theta}^*$, then there exists a co-state (adjoint) $\boldsymbol{p}^*$ which satisfies the backward differential equation*

$$\boldsymbol{m}^*_{t+1} = \boldsymbol{g}(\boldsymbol{m}^*_t; \boldsymbol{\theta}^*), \qquad \boldsymbol{m}^*_0 = [\boldsymbol{\chi}_{\mathcal{S}_k}; \mathbf{0}; \ldots; \mathbf{0}], \quad t = 0, \ldots, T-1, \tag{20a}$$

$$\boldsymbol{p}^*_t = \boldsymbol{p}^*_{t+1} \cdot \nabla_{\boldsymbol{m}} \boldsymbol{g}(\boldsymbol{m}^*_t; \boldsymbol{\theta}^*), \quad \boldsymbol{p}^*_T = -\nabla_{\boldsymbol{m}_T} \ell, \quad t = T-1, \ldots, 0. \tag{20b}$$

Moreover, the optimal $\boldsymbol{\theta}^*$ maximizes the total Hamiltonian: for any $\boldsymbol{\theta}$, there is

$$\textstyle\sum_{t=0}^{T-1} H(\boldsymbol{m}^*_t, \boldsymbol{p}^*_{t+1}; \boldsymbol{\theta}^*) \geq \sum_{t=0}^{T-1} H(\boldsymbol{m}^*_t, \boldsymbol{p}^*_{t+1}; \boldsymbol{\theta}). \tag{21}$$

In addition, for any given $\boldsymbol{\theta}$, there is $\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = -\sum_{t=0}^{T-1} \partial_{\boldsymbol{\theta}} H(\boldsymbol{m}^{\boldsymbol{\theta}}_t, \boldsymbol{p}^{\boldsymbol{\theta}}_{t+1}; \boldsymbol{\theta})$, where $\{\boldsymbol{m}^{\boldsymbol{\theta}}_t, \boldsymbol{p}^{\boldsymbol{\theta}}_t : 0 \leq t \leq T\}$ are obtained by the forward and backward passes (20a)-(20b) *with $\boldsymbol{\theta}$.*

The proof is given in Appendix B.4. We introduced the *total Hamiltonian* $\sum_{t=0}^{T-1} H(\boldsymbol{m}_t, \boldsymbol{p}_{t+1}; \boldsymbol{\theta})$ in Theorem 4 since the NMF dynamics (14) (or (16)) suggest a *time invariant* control $\boldsymbol{\theta}$ independent of t, which corresponds to $\boldsymbol{\theta}$ shared by all layers in an RNN. This is particularly important for time series analysis, where we perform regression on data observed within limited time window, but often want to use the learned parameters to predict events in distant future. Theorem 4 also implies that performing gradient descent to minimize $\mathcal{J}$ in (18a) with back-propagation is equivalent to maximizing the total Hamiltonian in light of (21).

Our numerical implementation of the proposed NMF is summarized in Algorithm 1. From training cascade data $\mathcal{D}$, NMF can learn the parameter $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{\eta}, \boldsymbol{w})$. The support (indices of nonzero entries) of the matrix $\boldsymbol{A}$ reveals the edge $\mathcal{E}$ of the diffusion network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and the values of $\boldsymbol{A}$ are the corresponding infection rates on the edges. In addition to the diffusion network parameters inferred by $\boldsymbol{A}$, we can also estimate (predict) the influence $\{\boldsymbol{x}_t : t \in [T]\}$ of any new source set $\boldsymbol{x}_0 \in \mathbb{R}^n$ by a forward pass of NMF dynamics (16) with the learned $\boldsymbol{\theta}$. Note that this forward pass can be computed on the fly, which is critical to those downstream applications (such as influence maximization) that call influence estimation as a subroutine repeatedly during the computations.

4 Numerical Experiments

Infection probability and influence function estimation We first test NMF on a set of synthetic networks that mimic the structure of real-world diffusion network. Two types of the Kronecker network model [27] are used: hierarchical (Hier) [8] and core-periphery (Core) [28] networks with parameter matrices [0.9,0.1;0.1,0.9] and [0.9,0.5;0.5,0.3], respectively. For each type of network model, we generate 5 networks consisting of 128 nodes and 512 edges. We simulate the diffusion where the infection times are modeled by exponential distribution (Exp) and Rayleigh distribution (Ray). For each distribution, we draw α_{ji} from Unif[0.1,1] to simulate the varying interactions between nodes. We generate training data consists of K=10,000 cascades, which is formed by 10 sample cascades for each of 1,000 source sets (a source set is generated by randomly selecting 1 to

Algorithm 1 Neural mean-field (NMF) algorithm for network inference and influence estimation

Input: $\mathcal{D} = \{\mathcal{C}_k : k \in [K]\}$ where $\mathcal{C}_k = \{\hat{\boldsymbol{x}}^{(k)}(t) \in \{0,1\}^n : t = 0, 1, \ldots, T\}$.
Initialization: Parameter $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{\eta}, \boldsymbol{w})$.
for $k = 1, \ldots, K$ **do**
  Sample a mini-batch $\hat{\mathcal{D}} \subset \mathcal{D}$ of cascades.
  Compute $\{\boldsymbol{m}_t : t \in [T]\}$ using (16) with $\boldsymbol{\theta}$ and $\boldsymbol{m}_0 = [\boldsymbol{\chi}_{\mathcal{S}}; \boldsymbol{0}]$ for each $\mathcal{C} \in \hat{\mathcal{D}}$. (Forward pass)
  Compute $\hat{\nabla}_{\boldsymbol{\theta}} \mathcal{J} = \sum_{\mathcal{C} \in \hat{\mathcal{D}}} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \hat{\boldsymbol{x}})$ with ℓ in (17). (Backward pass)
  Update parameter $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \tau \hat{\nabla}_{\boldsymbol{\theta}} \mathcal{J}$.
end for
Output: Network parameter $\boldsymbol{\theta}$.

10 nodes from the network). All networks and cascades are generated by SNAP [29]. Our numerical implementation of NMF is available at `https://github.com/ShushanHe/neural-mf`.

We compare NMF to two baseline methods: InfluLearner [12] which is a state-of-the-art method that learns the coverage function of each node for any fixed time, and a conditional LSTM (LSTM for short) [22], which are among the few existing methods capable of learning infection probabilities of individual nodes directly from cascade data as ours. For InfluLearner, we set 128 as the feature number for optimal accuracy as suggested in [12]. For LSTM, we use one LSTM block and a dense layer for each t. To evaluate accuracy, we compute the mean absolute error (MAE) of node infection probability and influence over 100 source sets for each time t. More details of the evaluation criteria are provided in Appendix D.1. The results are given in Figure 1, which shows the mean (center line) and standard deviation (shade) of the three methods. NMF generally has lowest MAE, except at some early stages where InfluLearner is better. Note that InfluLearner requires and benefits from the knowledge of the original source node for each infection in the cascade (provided in our training data), which is often unavailable in practice and not needed in our method.

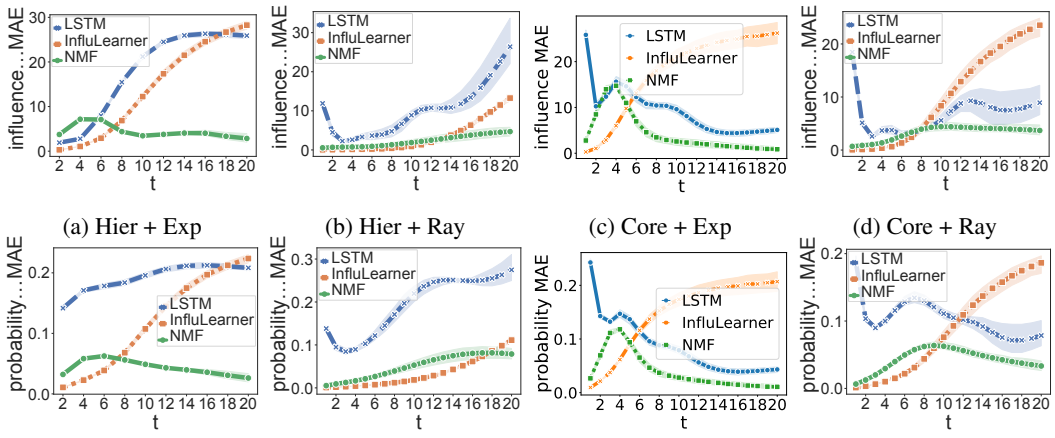
We also tested NMF on a real dataset [54] from Sina Weibo social platform consisting of more that 1.78 million users and 308 million following relationships among them. Following the setting in [12], we select the most popular tweet to generate diffusion cascades from the past 1,000 tweets of each user. Then we recreate the cascades by only keeping nodes of the top 1,000 frequency in the pooled node set over all cascades. For testing, we uniformly generate 100 source sets of size 10 and use $t = 1, 2, \ldots, 10$ as the time steps for observation. Finally, we test 100 source sets and compare our model NMF with the InfluLearner and LSTM. The MAE of all methods are shown in Figure 2a which shows that NMF significantly outperforms LSTM and is similar to InfluLearner. However, unlike InfluLearner that requires re-training for *every* t and is computationally expensive, NMF learns the evolution at all t in a single sweep of training and is tens of time faster.

We also test robustness of NMF for varying network density $|\mathcal{E}|/n$. The MAE of influence and infection probabilty by NMF on a hierarchical network with $n = 128$ are shown in Figure 2c and 2b, respectively. NMF remains accurate for denser networks, which can be notoriously difficult for other methods such as InfluLearner.

Network structure inference The interpretable parameterization of NMF allows us to explicitly learn the weight matrix $\boldsymbol{A}$. In this test, we examine the quality of the learned $\boldsymbol{A}$. We set the recovered adjacency matrix $\mathcal{E}$ to the binary indicator matrix $\boldsymbol{A}^\top \geq \epsilon$, i.e., $(\mathcal{E})_{i,j} = 1$ if $(\boldsymbol{A})_{ji} \geq 0.01$. To evaluate the quality of $\mathcal{E}$ and $\boldsymbol{A}$, we use four metrics: precision (Prc), recall (Rcl), accuracy (Acc), and correlation (Cor), defined as follows,

$$\text{Prc}(\mathcal{E}, \mathcal{E}^*) = \frac{|\mathcal{E} \cap \mathcal{E}^*|}{|\mathcal{E}^*|},\ \text{Rcl}(\mathcal{E}, \mathcal{E}^*) = \frac{|\mathcal{E} \cap \mathcal{E}^*|}{|\mathcal{E}|},\ \text{Acc}(\mathcal{E}, \mathcal{E}^*) = 1 - \frac{|\mathcal{E} - \mathcal{E}^*|}{|\mathcal{E}| + |\mathcal{E}^*|},\ \text{Cor}(A, A^*) = \frac{\text{tr}(A^\top A^*)}{\|A\|_F \|A^*\|_F}.$$

where $\mathcal{E}^*$ and $\boldsymbol{A}^*$ are their true values, respectively. In Acc, the edge set $\mathcal{E}$ is also interpreted as a matrix, and $|\mathcal{E}|$ counts the number of nonzeros in $\mathcal{E}$. In Cor, $\|A\|_F^2 = \text{tr}(A^\top A)$ is the Frobenius norm of the matrix A. Prc is the ratio of edges in $\mathcal{E}^*$ that are recovered in $\mathcal{E}$. Rcl is the ratio of correctly recovered edges in $\mathcal{E}$. Acc indicates the ratio of the number of common edges shared by $\mathcal{E}$ and $\mathcal{E}^*$ against the total number of edges in them. Cor measures similarity between A and A^* by taking their values into consideration. All metrics are bounded between $[0, 1]$, and higher value indicates better result. For comparison, we also test NETRATE [16] to the cascade data and learn $\boldsymbol{A}$ with Rayleigh distribution. Evaluation by four metrics are shown in Table 1, which indicates that



(a) Hier + Exp (b) Hier + Ray (c) Core + Exp (d) Core + Ray

(e) Hier + Exp (f) Hier + Ray (g) Core + Exp (h) Core + Ray

Figure 1: MAE of influence (top row) and node infection probability (bottom row) by LSTM, InfluLearner, and NMF on different combinations of Hierarchical (Hier) and Core-periphery (Core) networks, and exponential (Exp) and Rayleigh (Ray) diffusion models. Mean (centerline) and standard deviation (shade) over 100 tests are shown.

Table 1: Performance of structure inference using NETRATE and the proposed NMF on Random, Hierarchical, and Core-periphery networks with Rayleigh distribution as the diffusion time model on edges. Quality of the learned edge set $\mathcal{E}$ and distribution parameter $\boldsymbol{A}$ are measured by precision (Prc), recall (Rcl), accuracy (Acc), and correlation (Cor). Higher value indicates better quality.

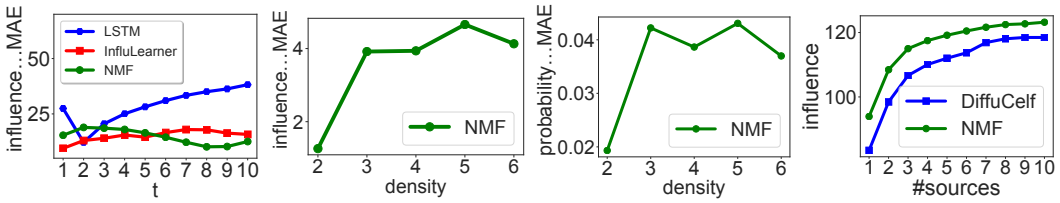
Network	Method	Prc	Rcl	Acc	Cor
Random	NETRATE	0.481	0.399	0.434	0.465
	NMF	**0.858**	**0.954**	**0.903**	**0.950**
Hierarchical	NETRATE	0.659	0.429	0.519	0.464
	NMF	**0.826**	**0.978**	**0.893**	**0.938**
Core-periphery	NETRATE	0.150	0.220	0.178	0.143
	NMF	**0.709**	**0.865**	**0.779**	**0.931**

NMF outperforms NETRATE in all metrics. Note that NMF learns $\boldsymbol{A}$ along with the NMF dynamics for infection probability estimation in its training, whereas NETRATE can only learn the matrix $\boldsymbol{A}$.

Influence maximization We use NMF as an influence estimation subroutine in a classical greedy algorithm [38] (NMF+Greedy), and compare with a state-of-the-art method DIFFCELF[42] for influence maximization (IM). Like NMF+Greedy, DIFFCELF also only requires infection time features, but not network structures as in most existing methods. We generate 1000 cascades with unique source (as required by DIFFCELF but not ours) on a hierarchical network of 128 nodes and 512 edges, and use exponential distribution for the transmission function with $\boldsymbol{A}$ generated from Unif[1,10]. Time window is $T = 20$. For each source set size $n_0 = 1, \ldots, 10$, NMF+Greedy and DIFFCELF are applied to identify the optimal source sets, whose influence are computed by averaging 10,000 MC simulated cascades. Figure 2d shows that the source sets obtained by NMF+Greedy generates greater influence than DIFFCELF consistently for every source size n_0.

5 Related Work

Sampling-based influence estimation methods have been considered for discrete-time and continuous-time diffusion models. Discrete-time models assume node infections only occur at discrete time points. Under this setting, the Independent Cascade (IC) model [24] is considered and a method with provable performance guarantee is developed for single source which iterates over a sequence of guesses of the true influence until the verifier accepts in [34]. To resolve the inefficiency of Monte



(a) Influ MAE vs t (b) Influ MAE vs density (c) Prob MAE vs density (d) Influence vs n_0

Figure 2: (a) MAE of influence estimated by LSTM, InfluLearner on Weibo data; (b)–(c) MAE of influence and infection probability of NMF for different network densities; (d) Influence of source sets selected by DIFFUCELF and NMF+Greedy for $n_0 = 1, \ldots, 10$.

Carlo simulations, the reverse influence sampling (RIS) sketching method [5] is adopted in [41]. Moreover, instead of using the full network structure, sketch-based approaches only characterize propagation instances for influence computation, such as the method in [10], which considers per-node summary structures defined by the bottom-k min-bash [9] sketch of the combined reachability set. In contrast to discrete-time models, continuous-time diffusion models allow arbitrary event occurrence times and hence are more accurate in modeling real-world diffusion processes. In Continuous-time Independent Cascade (CIC) models, influence estimation can be reformulated as the problem of finding the least label list which contains information about the distance to the smallest reachable labels from the source [13, 20]. Compared to methods using a fixed number of samples, a more scalable approximation scheme with a built-in block is developed to minimize the number of samples needed for the desired accuracy [40].

The aforementioned methods require knowledge of cascade traces [10] or the diffusion networks (review of related work on network structure inference is provided in Appendix C), such as node connectivity and node-to-node infection rates, as well as various assumptions on the diffusion of interests. However, such knowledge about the diffusion networks may not be available in practice, and the assumptions on the propagation or data formation are often application-specific and do not hold in most other problems. InfluLearner [12] is a state-of-the-art method that does not require knowledge of the underlying diffusion network. InfluLearner estimates the influence directly from cascades data in the CIC models by learning the influence function with a parameterization of the coverage functions using random basis functions. However, the random basis function suggested by [12] requires knowledge of the original source node for every infection, which can be difficult or impossible to be tracked in real-world applications.

In recent years, deep learning techniques have been employed to improve the scalability of influence estimation on large networks. In particular, convolutional neural networks (CNNs) and attention mechanism are incorporated with both network structures and user specific features to learn users' latent feature representation in [44]. By piping represented cascade graphs through a gated recurrent unit (GRU), the future incremental influence of a cascade can be predicted [31]. RNNs and CNNs are also applied to capture the temporal relationships on the user-generated contents networks (e.g., views, likes, comments, reposts) and extract more powerful features in [55]. In methods based on graph structures, graph neural networks (GNNs) and graph convolution networks (GCNs) are widely applied. In particular, two coupled GNNs are used to capture the interplay between node activation states and the influence spread [6], while GCNs integrated with teleport probability from the domain of page rank in [30] enhanced the performance of method in [44]. However, these methods depend critically on the structure or content features of cascades which may not be available in practice.

6 Conclusion

We proposed a novel framework using neural mean-field dynamics for inference and estimation on diffusion networks. Our new framework is derived from the Mori-Zwanzig formalism to obtain exact evolution of node infection probabilities. The memory term of the evolution can be approximated by convolutions, which renders the system as a delay differential equation and its time discretization reduces to a structured and interpretable RNN. Empirical study shows that our approach is versatile and robust to different variations of diffusion network models, and significantly outperforms existing approaches in accuracy and efficiency on both synthetic and real-world data sets.

Broader Impact

This paper makes a significant contribution to the learning of structure and infection probabilities for diffusion networks, which is one of the central problems in the study of stochastic information propagation on large heterogeneous networks. The proposed neural mean-field (NMF) dynamics provide the first principled approach for inference and estimation problems using cascade data. NMF is shown to be a delay differential equation with proper approximation of memory integral using learnable time convolution operators, and the system reduces to a highly structured and interpretable recurrent neural network after time discretiztion. Potential applications include influence maximization, outbreak detection, and source identification.

Acknowledgments and Disclosure of Funding

The work of SH and XY was supported in part by National Science Foundation under grants CMMI-1745382, DMS-1818886, and DMS-1925263. The work of HZ was supported in part by National Science Foundation IIS-1717916 and Shenzhen Research Institute of Big Data. He was on leave from College of Computing, Georgia Institute of Technology.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning, 2016.

[2] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[3] Á. Bodó, G. Y. Katona, and P. L. Simon. Sis epidemic propagation on hypergraphs. *Bulletin of mathematical biology*, 78(4):713–735, 2016.

[4] M. Boguná and R. Pastor-Satorras. Epidemic spreading in correlated complex networks. *Physical Review E*, 66(4):047104, 2002.

[5] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, Dec 2013.

[6] Q. Cao, H. Shen, J. Gao, B. Wei, and X. Cheng. Popularity prediction on social platforms with coupled graph neural networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, pages 70–78, New York, NY, USA, 2020. Association for Computing Machinery.

[7] A. J. Chorin, O. H. Hald, and R. Kupferman. Optimal prediction and the mori–zwanzig representation of irreversible processes. *Proceedings of the National Academy of Sciences*, 97(7):2968–2973, 2000.

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

[9] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55(3):441–453, 1997.

[10] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 629–638, New York, NY, USA, 2014. Association for Computing Machinery.

[11] X. Dong, D. Thanou, M. Rabbat, and P. Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

[12] N. Du, Y. Liang, M.-F. Balcan, and L. Song. Influence function learning in information diffusion networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–2016–II–2024. JMLR.org, 2014.

[13] N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha. Scalable influence estimation in continuous-time diffusion networks. In *Advances in Neural Information Processing Systems*, pages 3147–3155, 2013.

[14] N. Du, L. Song, M. Yuan, and A. J. Smola. Learning networks of heterogeneous influence. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2780–2788. Curran Associates, Inc., 2012.

[15] M. Farajtabar, X. Ye, S. Harati, L. Song, and H. Zha. Multistage campaigning in social networks. In *Advances in Neural Information Processing Systems*, pages 4718–4726, 2016.

[16] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. *arXiv preprint arXiv:1105.0697*, 2011.

[17] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 561–568, Madison, WI, USA, 2011. Omnipress.

[18] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):21, 2012.

[19] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf. Structure and dynamics of information pathways in online media. *CoRR*, abs/1212.1464, 2012.

[20] M. Gomez-Rodriguez, L. Song, N. Du, H. Zha, and B. Schölkopf. Influence estimation and maximization in continuous-time diffusion networks. *ACM Trans. Inf. Syst.*, 34(2), Feb. 2016.

[21] A. Gouasmi, E. J. Parish, and K. Duraisamy. A priori estimation of memory effects in reduced-order models of nonlinear systems using the mori-zwanzig formalism. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2205):20170385, 2017.

[22] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[23] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[24] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146. Association for Computing Machinery, 2003.

[25] D. Kempe, J. Kleinberg, and É. Tardos. Influential nodes in a diffusion model for social networks. In *Automata, languages and programming*, pages 1127–1138. Springer, 2005.

[26] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[27] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research*, 11:985–1042, 2010.

[28] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 695–704, New York, NY, USA, 2008. Association for Computing Machinery.

[29] J. Leskovec and R. Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

[30] C. K. Leung, A. Cuzzocrea, J. J. Mai, D. Deng, and F. Jiang. Personalized deepinf: Enhanced social influence prediction with deep learning and transfer learning. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2871–2880, 2019.

[31] C. Li, J. Ma, X. Guo, and Q. Mei. Deepcas: An end-to-end predictor of information cascades. In *Proceedings of the 26th international conference on World Wide Web*, pages 577–586, 2017.

[32] Q. Li, L. Chen, C. Tai, and E. Weinan. Maximum principle based algorithms for deep learning. *The Journal of Machine Learning Research*, 18(1):5998–6026, 2017.

[33] Y. Liang, Z. Jiang, and Y. Zheng. Inferring traffic cascading patterns. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '17, New York, NY, USA, 2017. Association for Computing Machinery.

[34] B. Lucier, J. Oren, and Y. Singer. Influence at scale: Distributed computation of complex contagion in networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 735–744, New York, NY, USA, 2015. Association for Computing Machinery.

[35] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro. Connecting the dots: Identifying network structure via graph signal processing. *IEEE Signal Processing Magazine*, 36(3):16–43, May 2019.

[36] J. C. Miller and I. Z. Kiss. Epidemic spread in networks: Existing methods and current challenges. *Mathematical modelling of natural phenomena*, 9(2):4, 2014.

[37] S. A. Myers and J. Leskovec. On the convexity of latent social network inference. *arXiv preprint arXiv:1010.5504*, 2010.

[38] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[39] M. Newman. *Networks: an introduction*. Oxford University Press, 2010.

[40] H. T. Nguyen, T. P. Nguyen, T. N. Vu, and T. N. Dinh. Outward influence and cascade size estimation in billion-scale networks. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(1), June 2017.

[41] N. Ohsaka, T. Akiba, Y. Yoshida, and K.-i. Kawarabayashi. Dynamic influence analysis in evolving networks. *Proc. VLDB Endow.*, 9(12):1077–1088, Aug. 2016.

[42] G. Panagopoulos, F. D. Malliaros, and M. Vazirgiannis. Diffugreedy: an influence maximization algorithm based on diffusion cascades. In *International Conference on Complex Networks and their Applications*, pages 392–404. Springer, 2018.

[43] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani. Epidemic processes in complex networks. *Reviews of modern physics*, 87(3):925, 2015.

[44] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2110–2119, 2018.

[45] M. G. Rodriguez and B. Schölkopf. Submodular inference of diffusion networks from multiple trees. *arXiv preprint arXiv:1205.1671*, 2012.

[46] Y. Rong, Q. Zhu, and H. Cheng. A model-free approach to infer the diffusion network from event cascade. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1653–1662, New York, NY, USA, 2016. Association for Computing Machinery.

[47] F. D. Sahneh and C. Scoglio. Epidemic spread in human networks. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 3008–3013. IEEE, 2011.

[48] P. Van Mieghem, J. Omic, and R. Kooij. Virus spread in networks. *Networking, IEEE/ACM Transactions on*, 17(1):1–14, 2009.

[49] M. Vergeer, L. Hermans, and S. Sams. Online social networks and micro-blogging in political campaigning the exploration of a new campaign tool and a new campaign style. *Party Politics*, 19(3):477–501, 2013.

[50] L. Wang, S. Ermon, and J. E. Hopcroft. Feature-enhanced probabilistic models for diffusion network inference. In *Machine Learning and Knowledge Discovery in Databases*, pages 499–514. Springer, 2012.

[51] Q. Wang, N. Ripamonti, and J. S. Hesthaven. Recurrent neural network closure of parametric pod-galerkin reduced-order models based on the mori-zwanzig formalism. *Journal of Computational Physics*, 410:109402, 2020.

[52] J. Wortman. Viral marketing and the diffusion of trends on social networks. *Tech Report*, 2008.

[53] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2020.

[54] J. Zhang, B. Liu, J. Tang, T. Chen, and J. Li. Social influence locality for modeling retweeting behaviors. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2761–2767. AAAI Press, 2013.

[55] Y. Zhu, J. Xie, and Z. Chen. Predicting the popularity of micro-videos with multimodal variational encoder-decoder framework, 2020.

A An Illustrative Example of Mori-Zwanzig Formalism

The following problem is widely used as an introductory example of the Mori-Zwanzig (MZ) formalism [21, 51] for model order reduction: let $\boldsymbol{z} = [\boldsymbol{x}; \boldsymbol{y}] \in \mathbb{R}^N$ where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^{N-n}$, and $n \ll N$. Consider the system of linear differential equations

$$\begin{cases} \boldsymbol{x}' &= \boldsymbol{A}_{11}\boldsymbol{x} + \boldsymbol{A}_{12}\boldsymbol{y}, \\ \boldsymbol{y}' &= \boldsymbol{A}_{21}\boldsymbol{x} + \boldsymbol{A}_{22}\boldsymbol{y}, \end{cases} \tag{22}$$

with initial values $\boldsymbol{x}(0) = \boldsymbol{x}_0$ and $\boldsymbol{y}(0) = \boldsymbol{y}_0$. Suppose we are interested in the time evolution of $\boldsymbol{x}(t)$, which depends on the joint effect of $\boldsymbol{x}$ and $\boldsymbol{y}$. However the computation of the complete system (22) is expensive and can be prohibitive for large N. The question is whether we can derive a reduced system only involving $\boldsymbol{x}$ from (22). To this end, we assume $\boldsymbol{x}$ is given, and solve for $\boldsymbol{y}$ from the $\boldsymbol{y}$-equation of (22) to obtain

$$\boldsymbol{y}(t) = e^{\boldsymbol{A}_{22}t}\boldsymbol{y}_0 + \int_0^t e^{\boldsymbol{A}_{22}(t-s)}\boldsymbol{A}_{22}x(s)\,\mathrm{d}s. \tag{23}$$

Then we plug this back into the $\boldsymbol{x}$-equation of (22) and obtain

$$\boldsymbol{x}'(t) = \boldsymbol{A}_{11}\boldsymbol{x}(t) + \boldsymbol{A}_{12}\int_0^t e^{\boldsymbol{A}_{22}(t-s)}\boldsymbol{A}_{22}\boldsymbol{x}(s)\,\mathrm{d}s + \boldsymbol{A}_{12}e^{\boldsymbol{A}_{22}t}\boldsymbol{y}_0, \tag{24}$$

which neglects the dependence on $\boldsymbol{y}(t)$ except for the initial value $\boldsymbol{y}_0$.

As shown in the example above, MZ formalism aims at reducing a high-dimensional system of $\boldsymbol{z}$ into a low-dimensional system of $\boldsymbol{x}$ (resolved variable) while maintaining the effect of $\boldsymbol{y}$ (unresolved variable). This is particularly useful if an exact solution of $\boldsymbol{z}$ is unnecessary to understand the dynamics of $\boldsymbol{x}$. Specialized derivations and subsequent approximation techniques can be implemented to obtain highly efficient numerical solutions for nonlinear systems.

B Proofs

B.1 Proof of Theorem 1

Proof. Let $\lambda_i^*(t)$ be the conditional intensity of node i at time t, i.e., $\mathbb{E}[\mathrm{d}X_i(t)|\mathcal{H}(t)] = \lambda_i^*(t)\,\mathrm{d}t$. In the standard diffusion model, the conditional intensity $\lambda_i^*(t)$ of a healthy node i (i.e., $X_i(t) = 0$) is determined by the total infection rate of its infected neighbors j (i.e., $X_j(t) = 1$). That is,

$$\lambda_i^*(t) = \sum_j \alpha_{ji}X_j(t)(1 - X_i(t)). \tag{25}$$

By taking expectation $\mathbb{E}_{\mathcal{H}(t)}[\cdot]$ on both sides of (25), we obtain

$$\begin{aligned} \lambda_i(t) :=& \mathbb{E}_{\mathcal{H}(t)}[\lambda_i^*(t)] = \mathbb{E}_{\mathcal{H}(t)}\left[\alpha_{ji}X_j(t)(1 - X_i(t))\big|\mathcal{H}(t)\right] \\ &= \sum_j \alpha_{ji}(x_j - x_{ij}) = \sum_j \alpha_{ji}(x_j - y_{ij} - e_{ij}). \end{aligned} \tag{26}$$

On the other hand, there is

$$\lambda_i(t)\,\mathrm{d}t = \mathbb{E}_{\mathcal{H}(t)}[\lambda_i^*(t)]\,\mathrm{d}t = \mathbb{E}_{\mathcal{H}(t)}[\mathrm{d}X_i(t)|\mathcal{H}(t)] = \mathrm{d}\mathbb{E}_{\mathcal{H}(t)}[X_i(t)|\mathcal{H}(t)] = \mathrm{d}x_i. \tag{27}$$

Combining (26) and (27) yields

$$x_i' = \frac{\mathrm{d}x_i(t)}{\mathrm{d}t} = \sum_j \alpha_{ji}(x_j - y_{ij} - e_{ij}) = (\boldsymbol{A}\boldsymbol{x})_i - (\mathrm{diag}(\boldsymbol{x})\boldsymbol{A}\boldsymbol{x})_i - \sum_j \alpha_{ji}e_{ij}$$

for every $i \in [n]$, which verifies the $\boldsymbol{x}$ part of (6). Similarly, we can obtain

$$x_I' = \sum_{i\in I}\sum_{j\notin I}\alpha_{ji}(x_I - x_{I\cup\{j\}}) = \sum_{i\in I}\sum_{j\notin I}\alpha_{ji}(y_I + e_I - y_{I\cup\{j\}} - e_{I\cup\{j\}}). \tag{28}$$

Moreover, by taking derivative on both sides of $x_I(t) = y_I(t) + e_I(t)$, we obtain

$$x_I' = \sum_{i\in I} y_{I\setminus\{i\}} x_i' + e_I' = \sum_{i\in I} y_{I\setminus\{i\}} \sum_{j\neq i} \alpha_{ji}(x_j - x_i x_j - e_{ij}) + e_I'. \tag{29}$$

Combining (28) and (29) yields the $\boldsymbol{e}$ part of (6).

It is clear that $\boldsymbol{x}_0 = \boldsymbol{\chi}_{\mathcal{S}}$. For every I, at time $t = 0$, there is $x_I(0) = \prod_{i\in I} X_i(0) = 1$ if $I \subset \mathcal{S}$ and 0 otherwise; and the same for $y_I(0)$. Hence $e_I(0) = x_I(0) - y_I(0) = 0$ for all I. Hence $\boldsymbol{z}_0 = [\boldsymbol{x}_0; \boldsymbol{e}_0] = [\boldsymbol{\chi}_{\mathcal{S}}; \mathbf{0}]$, which verifies the initial condition of (6). □

B.2 Proof of Theorem 2

Proof. Consider the system (6) over a finite time horizon $[0, T]$, which evolves on a smooth manifold $\Gamma \subset \mathbb{R}^N$. For any real-valued phase (observable) space function $g : \Gamma \to \mathbb{R}$, the nonlinear system (6) is equivalent to the linear partial differential equation, known as the Liouville equation:

$$\begin{cases} \partial_t u(t, \boldsymbol{z}) = \mathcal{L}[u](t, \boldsymbol{z}), \\ u(0, \boldsymbol{z}) = g(\boldsymbol{z}), \end{cases} \tag{30}$$

where the Liouville operator $\mathcal{L}[u] := \bar{\boldsymbol{f}}(\boldsymbol{z}) \cdot \nabla_{\boldsymbol{z}} u$. The equivalency is in the sense that the solution of (30) satisfies $u(t, \boldsymbol{z}_0) = g(\boldsymbol{z}(t; \boldsymbol{z}_0))$, where $\boldsymbol{z}(t; \boldsymbol{z}_0)$ is the solution to (6) with initial value $\boldsymbol{z}_0$.

Denote $e^{t\mathcal{L}}$ the Koopman operator associated with $\mathcal{L}$ such that $e^{t\mathcal{L}} g(\boldsymbol{z}_0) = g(\boldsymbol{z}(t))$ where $\boldsymbol{z}(t)$ is the solution of (6). Then $e^{t\mathcal{L}}$ satisfies the semi-group property, i.e.,

$$e^{t\mathcal{L}} g(z) = g(e^{t\mathcal{L}} z) \tag{31}$$

for all g. On the right hand side of (31), $\boldsymbol{z}$ can be interpreted as $\boldsymbol{z} = \boldsymbol{\iota}(\boldsymbol{z}) = [\iota_1(\boldsymbol{z}), \ldots, \iota_N(\boldsymbol{z})]$ where $\iota_j(\boldsymbol{z}) = z_j$ for all j.

Now consider the projection operator $\mathcal{P}$ as the truncation such that $\mathcal{P} g(\boldsymbol{z}) = \mathcal{P} g(\boldsymbol{x}, \boldsymbol{e}) = g(\boldsymbol{x}, 0)$ for any $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{e})$, and its orthogonal complement as $\mathcal{Q} = I - \mathcal{P}$ where I is the identity operator. Note that $\boldsymbol{z}'(t) = \frac{\mathrm{d}\boldsymbol{z}(t)}{\mathrm{d}t} = \frac{\partial}{\partial t} e^{t\mathcal{L}} \boldsymbol{z}_0$, and $\bar{\boldsymbol{f}}(\boldsymbol{z}(t)) = e^{t\mathcal{L}} \boldsymbol{f}(\boldsymbol{z}_0) = e^{t\mathcal{L}} \mathcal{L} \boldsymbol{z}_0$ since $\mathcal{L} \iota_j(\boldsymbol{z}) = \boldsymbol{f}_j(\boldsymbol{z})$ for all $\boldsymbol{z}$ and j. Therefore (6) implies that

$$\frac{\partial}{\partial t} e^{t\mathcal{L}} \boldsymbol{z}_0 = e^{t\mathcal{L}} \mathcal{L} \boldsymbol{z}_0 = e^{t\mathcal{L}} \mathcal{P}\mathcal{L} \boldsymbol{z}_0 + e^{t\mathcal{L}} \mathcal{Q}\mathcal{L} \boldsymbol{z}_0. \tag{32}$$

Note that the first term on the right hand side of (32) is

$$e^{t\mathcal{L}} \mathcal{P}\mathcal{L} \boldsymbol{z}_0 = \mathcal{P}\mathcal{L} e^{t\mathcal{L}} \boldsymbol{z}_0 = \mathcal{P}\mathcal{L} \boldsymbol{z}(t). \tag{33}$$

For the second term in (32), we recall that the well-known Dyson's identity for the Koopman operator $\mathcal{L}$ is given by

$$e^{t\mathcal{L}} = e^{t\mathcal{Q}\mathcal{L}} + \int_0^t e^{s\mathcal{L}} \mathcal{P}\mathcal{L} e^{(t-s)\mathcal{Q}\mathcal{L}} \,\mathrm{d}s. \tag{34}$$

Applying (34) to $\mathcal{Q}\mathcal{L}\boldsymbol{z}_0$ yields

$$\begin{aligned} e^{t\mathcal{L}} \mathcal{Q}\mathcal{L} \boldsymbol{z}_0 &= e^{t\mathcal{Q}\mathcal{L}} \mathcal{Q}\mathcal{L} \boldsymbol{z}_0 + \int_0^t e^{s\mathcal{L}} \mathcal{P}\mathcal{L} e^{(t-s)\mathcal{Q}\mathcal{L}} \mathcal{Q}\mathcal{L} \boldsymbol{z}_0 \,\mathrm{d}s \\ &= e^{t\mathcal{Q}\mathcal{L}} \mathcal{Q}\mathcal{L} \boldsymbol{z}_0 + \int_0^t \mathcal{P}\mathcal{L} e^{(t-s)\mathcal{Q}\mathcal{L}} \mathcal{Q}\mathcal{L} e^{s\mathcal{L}} \boldsymbol{z}_0 \,\mathrm{d}s \\ &= e^{t\mathcal{Q}\mathcal{L}} \mathcal{Q}\mathcal{L} \boldsymbol{z}_0 + \int_0^t \mathcal{P}\mathcal{L} e^{(t-s)\mathcal{Q}\mathcal{L}} \mathcal{Q}\mathcal{L} \boldsymbol{z}(s) \,\mathrm{d}s. \end{aligned} \tag{35}$$

Substituting (33) and (35) into (32), we obtain

$$\frac{\partial}{\partial t} e^{t\mathcal{L}} \boldsymbol{z}_0 = \mathcal{P}\mathcal{L} \boldsymbol{z}(t) + e^{t\mathcal{Q}\mathcal{L}} \mathcal{Q}\mathcal{L} \boldsymbol{z}_0 + \int_0^t \mathcal{P}\mathcal{L} e^{(t-s)\mathcal{Q}\mathcal{L}} \mathcal{Q}\mathcal{L} \boldsymbol{z}(s) \,\mathrm{d}s, \tag{36}$$

where we used the fact that $e^{t\mathcal{L}} \mathcal{P}\mathcal{L} \boldsymbol{z}_0 = \mathcal{P}\mathcal{L} e^{t\mathcal{L}} \boldsymbol{z}_0 = \mathcal{P}\mathcal{L} \boldsymbol{z}(t)$. Denote $\boldsymbol{\phi}(t, \boldsymbol{z}) := e^{t\mathcal{L}} \mathcal{Q}\mathcal{L} \boldsymbol{z}$, then we simplify (36) into

$$\frac{\partial}{\partial t} e^{t\mathcal{L}} \boldsymbol{z}_0 = \mathcal{P}\mathcal{L} \boldsymbol{z}(t) + \boldsymbol{\phi}(t, \boldsymbol{z}_0) + \int_0^t \boldsymbol{k}(t - s, \boldsymbol{z}(s)) \,\mathrm{d}s, \tag{37}$$

where $\boldsymbol{k}(t, \boldsymbol{z}) := \mathcal{P}\mathcal{L}\boldsymbol{\phi}(t, \boldsymbol{z}) = \mathcal{P}\mathcal{L}e^{t\mathcal{L}}\mathcal{Q}\mathcal{L}\boldsymbol{z}$.

Now consider the evolution of $\boldsymbol{\phi}(t, \boldsymbol{z})$, which is given by

$$\partial_t \boldsymbol{\phi}(t, \boldsymbol{z}_0) = \mathcal{Q}\mathcal{L}\boldsymbol{\phi}(t, \boldsymbol{z}_0), \tag{38}$$

with initial condition $\boldsymbol{\phi}(0, \boldsymbol{z}_0) = \mathcal{Q}\mathcal{L}\boldsymbol{z}_0 = \mathcal{L}\boldsymbol{z}_0 - \mathcal{P}\mathcal{L}\boldsymbol{z}_0 = \bar{\boldsymbol{f}}(\boldsymbol{x}_0, \boldsymbol{e}_0) - \bar{\boldsymbol{f}}(\boldsymbol{x}_0, \boldsymbol{0}) = \boldsymbol{0}$ since $\boldsymbol{e}_0 = \boldsymbol{0}$. Applying $\mathcal{P}$ on both sides of (38) yields

$$\partial_t \mathcal{P}\boldsymbol{\phi}(t, \boldsymbol{z}_0) = \mathcal{P}\mathcal{Q}\mathcal{L}\boldsymbol{\phi}(t, \boldsymbol{z}_0) = \boldsymbol{0},$$

with initial $\mathcal{P}\boldsymbol{\phi}(0, \boldsymbol{z}_0) = \boldsymbol{0}$. This implies that $\mathcal{P}\boldsymbol{\phi}(t, \boldsymbol{z}_0) = \boldsymbol{0}$ for all t. Hence, applying $\mathcal{P}$ to both sides of (36) yields

$$\frac{\partial}{\partial t}\mathcal{P}\boldsymbol{z}(t) = \frac{\partial}{\partial t}\mathcal{P}e^{t\mathcal{L}}\boldsymbol{z}_0 = \mathcal{P}\mathcal{L}\boldsymbol{z}(t) + \int_0^t \mathcal{P}\boldsymbol{k}(t-s, \boldsymbol{z}(s))\,\mathrm{d}s. \tag{39}$$

Restricting to the first n components, $\mathcal{P}\boldsymbol{z}(t)$ reduces to $\boldsymbol{x}(t)$ and $\mathcal{P}\boldsymbol{k}(t-s, \boldsymbol{z}(s))$ reduces to $\boldsymbol{k}(t-s, \boldsymbol{x}(s))$. Recalling that $\mathcal{P}\mathcal{L}\boldsymbol{z}(t) = \mathcal{P}\bar{\boldsymbol{f}}(\boldsymbol{z}(t)) = \bar{\boldsymbol{f}}(\boldsymbol{x}(t), \boldsymbol{0}) = \boldsymbol{f}(\boldsymbol{x}(t))$ completes the proof. □

B.3 Proof of Theorem 3

Proof. From the definition of $\boldsymbol{h}(t)$ in (40), we obtain

$$\boldsymbol{h} = \int_0^t \boldsymbol{K}(t-s; \boldsymbol{w})\boldsymbol{x}(s)\,\mathrm{d}s = \int_{-\infty}^t \boldsymbol{K}(t-s; \boldsymbol{w})\boldsymbol{x}(s)\,\mathrm{d}s = \int_0^\infty \boldsymbol{K}(s; \boldsymbol{w})\boldsymbol{x}(t-s)\,\mathrm{d}s \tag{40}$$

where we used the fact that $\boldsymbol{x}(t) = 0$ for $t < 0$. Taking derivative on both sides of (40) yields

$$\begin{aligned}\boldsymbol{h}' &= \int_0^\infty \boldsymbol{K}(s; \boldsymbol{w})\boldsymbol{x}'(t-s)\,\mathrm{d}s = \int_0^\infty \boldsymbol{K}(s; \boldsymbol{w})\tilde{\boldsymbol{f}}(\boldsymbol{x}(t-s), \boldsymbol{h}(t-s); \boldsymbol{A}, \boldsymbol{\eta})\,\mathrm{d}s \\ &= \int_{-\infty}^t \boldsymbol{K}(t-s; \boldsymbol{w})\tilde{\boldsymbol{f}}(\boldsymbol{x}(s), \boldsymbol{h}(s); \boldsymbol{A}, \boldsymbol{\eta})\,\mathrm{d}s = \int_0^t \boldsymbol{K}(t-s; \boldsymbol{w})\tilde{\boldsymbol{f}}(\boldsymbol{x}(s), \boldsymbol{h}(s); \boldsymbol{A}, \boldsymbol{\eta})\,\mathrm{d}s\end{aligned}$$

where we used the fact that $\boldsymbol{x}'(t) = \tilde{\boldsymbol{f}}(\boldsymbol{x}(t), \boldsymbol{h}(t); \boldsymbol{A}, \boldsymbol{\eta}) = 0$ for $t < 0$ in the last equality.

If $\boldsymbol{K}(t; \boldsymbol{w}) = \sum_l \boldsymbol{B}_l e^{-\boldsymbol{C}_l t}$, then we can take derivative of (40) and obtain

$$\begin{aligned}\boldsymbol{h}'(t) &= \sum_{l=1}^L \frac{\mathrm{d}}{\mathrm{d}t}\Big(\int_{-\infty}^t \boldsymbol{B}_l e^{-\boldsymbol{C}_l t}\boldsymbol{x}(s)\,\mathrm{d}s\Big) = \sum_{l=1}^L \Big(\boldsymbol{B}_l\boldsymbol{x}(t) - \int_{-\infty}^t \boldsymbol{B}_l\boldsymbol{C}_l e^{-\boldsymbol{C}_l t}\boldsymbol{x}(s)\,\mathrm{d}s\Big) \\ &= \sum_{l=1}^L \Big(\boldsymbol{B}_l\boldsymbol{x}(t) - \boldsymbol{C}_l\int_{-\infty}^t \boldsymbol{B}_l e^{-\boldsymbol{C}_l t}\boldsymbol{x}(s)\,\mathrm{d}s\Big) = \sum_{l=1}^L (\boldsymbol{B}_l\boldsymbol{x}(t) - \boldsymbol{C}_l\boldsymbol{h}(t)).\end{aligned}$$

Time discretization (14) can then be obtained by finite difference in time with normalized step size 1 and proper scaling of the network parameters $\boldsymbol{\theta}$. □

B.4 Proof of Theorem 4

Proof. We consider the augmented state $\boldsymbol{\xi}$ and nonlinear dynamics $\bar{\boldsymbol{g}}(\cdot; \boldsymbol{\theta})$ associated with $\boldsymbol{m}$ and $\boldsymbol{g}(\cdot; \boldsymbol{\theta})$, defined as follows:

$$\boldsymbol{\xi}_0 = \begin{bmatrix}\boldsymbol{m}_0 \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0}\end{bmatrix}, \quad \boldsymbol{\xi}_1 = \bar{\boldsymbol{g}}(\boldsymbol{\xi}_0; \boldsymbol{\theta}) := \begin{bmatrix}\boldsymbol{g}^1(\boldsymbol{m}_0; \boldsymbol{\theta}) \\ \boldsymbol{g}^2(\boldsymbol{m}_0; \boldsymbol{\theta}) \\ \vdots \\ \boldsymbol{g}^T(\boldsymbol{m}_0; \boldsymbol{\theta})\end{bmatrix} = \begin{bmatrix}\boldsymbol{m}_1 \\ \boldsymbol{m}_2 \\ \vdots \\ \boldsymbol{m}_T\end{bmatrix}, \tag{41}$$

where $\boldsymbol{g}^t$ stands for the composition of $\boldsymbol{g}(\cdot; \boldsymbol{\theta})$ for t times.

Without overloading the notations, we reuse $\mathcal{J}$ and ℓ of the objective function (18a) and loss function (17) of $\boldsymbol{m}$ respectively for the augmented state $\boldsymbol{\xi}$. In addition, following [32], we further simpify the notation by combining the K training data into a single variable $\hat{\boldsymbol{x}} := [\hat{\boldsymbol{x}}^{(1)}, \ldots, \hat{\boldsymbol{x}}^{(K)}]$; similar

for the state variable $\boldsymbol{x}$. In this case, the dynamics $\boldsymbol{g}$ is applied to each column of $\boldsymbol{x}$, and the loss function ℓ is to be interpreted as the average loss as in (17). Furthermore, we temporarily assume the regularization $r(\boldsymbol{\theta}) = 0$ as it is simple to append $\boldsymbol{\theta}$ to the state $\boldsymbol{\xi}$ and merge $r(\boldsymbol{\theta})$ into the loss function $\ell(\boldsymbol{\xi}, \hat{\boldsymbol{\xi}})$. Then the optimal control problem (18) is rewritten as

$$\min_{\boldsymbol{\theta}} \quad \mathcal{J}(\boldsymbol{\theta}) := \ell(\boldsymbol{\xi}, \hat{\boldsymbol{\xi}}) + r(\boldsymbol{\theta}) \tag{42a}$$

$$\text{s.t.} \quad \boldsymbol{\xi}_1 = \bar{\boldsymbol{g}}(\boldsymbol{\xi}_0; \boldsymbol{\theta}), \quad \boldsymbol{\xi}_0 = [\boldsymbol{m}_0; \mathbf{0}; \ldots; \mathbf{0}]. \tag{42b}$$

Note that (42) is a one-step optimal control with $\bar{\boldsymbol{g}}(\cdot; \boldsymbol{\theta})$. Now by the discrete Pontryagin's Maximum Principle [2], for the state $\boldsymbol{\xi}^*$ optimally controlled by $\boldsymbol{\theta}^*$, there exists a co-state $\boldsymbol{\psi}^*$, such that $\boldsymbol{\xi}^*$ and $\boldsymbol{\psi}^*$ satisfy the following forward and backward equations for $\boldsymbol{\theta} = \boldsymbol{\theta}^*$:

$$\boldsymbol{\xi}_1^* = \bar{\boldsymbol{g}}(\boldsymbol{\xi}_0^*; \boldsymbol{\theta}^*), \qquad \boldsymbol{\xi}_0^* = [\boldsymbol{m}_0; \mathbf{0}; \ldots; \mathbf{0}], \tag{43a}$$

$$\boldsymbol{\psi}_0^* = \boldsymbol{\psi}_1^* \cdot \nabla_{\boldsymbol{\xi}} \bar{\boldsymbol{g}}(\boldsymbol{\xi}_1^*; \boldsymbol{\theta}^*), \quad \boldsymbol{\psi}_1^* = -\nabla_{\boldsymbol{\xi}} \ell(\boldsymbol{\xi}_1^*, \hat{\boldsymbol{\xi}}), \tag{43b}$$

where

$$\boldsymbol{\xi}_1^* = [\boldsymbol{m}_1^*; \ldots; \boldsymbol{m}_T^*] \quad \text{and} \quad \boldsymbol{\psi}_1^* = [\partial_{\boldsymbol{m}_1} \ell(\boldsymbol{\xi}_1^*, \hat{\boldsymbol{\xi}}); \ldots; \partial_{\boldsymbol{m}_T} \ell(\boldsymbol{\xi}_1^*, \hat{\boldsymbol{\xi}})] = [\boldsymbol{p}_1^*; \ldots; \boldsymbol{p}_T^*]. \tag{44}$$

In addition, $\boldsymbol{\theta}^*$ maximizes the Hamiltonian $\mathcal{H}$ associated with (43):

$$\mathcal{H}(\boldsymbol{\xi}^*, \boldsymbol{\psi}^*; \boldsymbol{\theta}^*) \geq \mathcal{H}(\boldsymbol{\xi}^*, \boldsymbol{\psi}^*; \boldsymbol{\theta}), \ \forall\, \boldsymbol{\theta}, \quad \text{where} \quad \mathcal{H}(\boldsymbol{\xi}, \boldsymbol{\psi}; \boldsymbol{\theta}) := \boldsymbol{\psi}_1 \cdot \bar{\boldsymbol{g}}(\boldsymbol{\xi}_0; \boldsymbol{\theta}) - r(\boldsymbol{\theta}). \tag{45}$$

Combining (44), (45), and the definition of H in (19) yields the maximization of total Hamiltonian at the optimal control $\boldsymbol{\theta}^*$:

$$\textstyle\sum_{t=0}^{T-1} H(\boldsymbol{m}_t^*, \boldsymbol{p}_{t+1}^*; \boldsymbol{\theta}^*) \geq \sum_{t=0}^{T-1} H(\boldsymbol{m}_t^*, \boldsymbol{p}_{t+1}^*; \boldsymbol{\theta}), \quad \forall\, \boldsymbol{\theta}.$$

For any control $\boldsymbol{\theta}$ and its state and co-state variables $\boldsymbol{\xi}^{\boldsymbol{\theta}}$ and $\boldsymbol{\psi}^{\boldsymbol{\theta}}$ following (43) with $\boldsymbol{\theta}$ (also corresponding to $\boldsymbol{m}_t^{\boldsymbol{\theta}}$ and $\boldsymbol{p}_t^{\boldsymbol{\theta}}$ for $t = 0, \ldots, T$), we have

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\xi}} \ell(\boldsymbol{\xi}_1^{\boldsymbol{\theta}}, \hat{\boldsymbol{\xi}}) \cdot \nabla_{\boldsymbol{\theta}} \boldsymbol{\xi}_1^{\boldsymbol{\theta}} + \nabla_{\boldsymbol{\theta}} r(\boldsymbol{\theta}) \\
&= [\partial_{\boldsymbol{m}_1} \ell(\boldsymbol{\xi}_1^{\boldsymbol{\theta}}, \hat{\boldsymbol{\xi}}); \ldots; \partial_{\boldsymbol{m}_T} \ell(\boldsymbol{\xi}_1^{\boldsymbol{\theta}}, \hat{\boldsymbol{\xi}})] \cdot [\partial_{\boldsymbol{\theta}} \boldsymbol{g}(\boldsymbol{m}_0^{\boldsymbol{\theta}}; \boldsymbol{\theta}); \ldots; \partial_{\boldsymbol{\theta}} \boldsymbol{g}(\boldsymbol{m}_{T-1}^{\boldsymbol{\theta}}; \boldsymbol{\theta})] + \nabla_{\boldsymbol{\theta}} r(\boldsymbol{\theta}) \\
&= -\textstyle\sum_{t=1}^{T} \left(\boldsymbol{p}_t^{\boldsymbol{\theta}} \cdot \partial_{\boldsymbol{\theta}} \boldsymbol{g}(\boldsymbol{m}_t^{\boldsymbol{\theta}}; \boldsymbol{\theta}) + \frac{1}{T} \nabla_{\boldsymbol{\theta}} r(\boldsymbol{\theta}) \right) \\
&= -\textstyle\sum_{t=1}^{T} \partial_{\boldsymbol{\theta}} H(\boldsymbol{m}_t^{\boldsymbol{\theta}}, \boldsymbol{p}_{t+1}^{\boldsymbol{\theta}}; \boldsymbol{\theta}),
\end{aligned}$$

which completes the proof. □

C Additional Related Work

Network structure inference Inference of diffusion network structure is an important problem closely related to influence estimation. In particular, if the network structure and infections rates are unknown, one often needs to first infer such information from a training dataset of sampled cascades, each of which tracks a series of infection times and locations on the network. Existing methods have been proposed to infer network connectivity [18, 45, 33, 14] and also the infection rates between nodes [37, 17, 19]. Submodular optimization is applied to infer network connectivity [18, 45, 33] by considering the most probable [18] or all [45, 33] directed trees supported by each cascade. One of the early works that incorporate spatio-temporal factors into network inference is introduced in [33]. Utilizing convex optimization, transmission functions [14], the prior probability [37], and the transmission rate [17] over edges are inferred from cascades. In addition to static networks, the infection rates are considered but also in the unobserved dynamic network changing over time [19]. Besides cascades, other features of dynamical processes on networks have been used to infer the diffusion network structures. To avoid using predefined transmission models, the statistical difference of the infection time intervals between nodes in the same cascade versus those not in any cascade was considered in [46]. A given time series of the epidemic prevalence, i.e., the average fraction of infected nodes was applied to discover the underlying network. The recurrent cascading behavior is also explained by integrating a feature vector describing the additional features [50]. A graph signal processing (GSP) approach is developed to infer graph structure from dynamics on networks [35, 11].

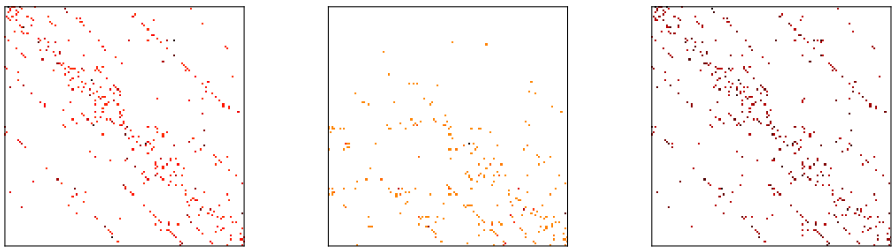
D Experiment Supplements

D.1 Implementation details

In our NMF implementation, we use a standard LSTM architecture and 3 dense layers for the RNN ε at each time t. Regularization terms using l_1-norm of all parameters are added to the loss function to promote their sparsity and robustness. Specifically, we use 0.001 to weight $\boldsymbol{A}$ and 0.0001 to all other trainable parameters, respectively. The NMF networks are trained and tested in TensorFlow [1] using Adam optimizer with default parameters (lr=0.001, β_1=0.9, β_2=0.999, ϵ=1e-8) on a Linux workstation with Intel 8-Core Turbo 5GHz CPU, 64GB of memory, and an Nvidia RTX 2080Ti GPU. The LSTM model is trained and tested in the same setting as NMF except a fixed regularization weight 0.001 for all trainable parameters. InfluLearner is trained in Matlab, and the number of features is set to 128. All experiments are performed on the same machine. Given ground truth node infection probability $\boldsymbol{x}^*$, the Mean Absolute Error (MAE) of influence (Inf) and infection probability (Prob) of estimated $\boldsymbol{x}$ are defined by $|\mathbf{1} \cdot (\boldsymbol{x}_t - \boldsymbol{x}_t^*)|$ and $\|\boldsymbol{x}_t - \boldsymbol{x}_t^*\|_1/n$ for every t, respectively.

D.2 Inference of node interdependencies

Due to its highly interpretable structure, NMF can also learn the node inter-dependencies through $\boldsymbol{A}$. In addition to the quantitative evaluations provided in Section 4, we show the visual appearance of $\boldsymbol{A}$ inferred by NMF in Figure 3. The ground truth $\boldsymbol{A}^*$ and $\boldsymbol{A}$ inferred by NETRATE are also provided for comparison. As we can see, $\boldsymbol{A}$ inferred by NMF is much more faithful to $\boldsymbol{A}^*$ than that by NETRATE. Note that NETRATE requires knowledge of specific diffusion model type (Rayleigh in this test) whereas NMF does not. This result shows that NMF is versatile and robust when only cascade data are available.



(a) True (b) NETRATE (c) NMF

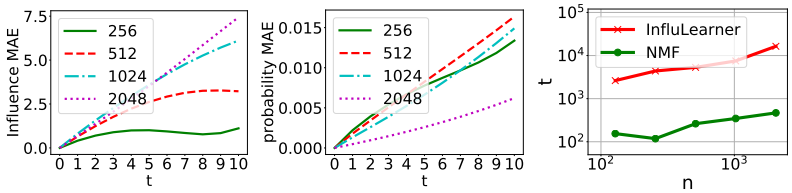
Figure 3: Ground truth $\boldsymbol{A}^*$ (left) and $\boldsymbol{A}$ inferred NETRATE (middle) and NMF (right) under the same color scale using cascaded data from a Hierarchical network with Rayleigh diffusion model.

D.3 Accuracy and Scalability

Accuracy for networks of increasing sizes We test NMF on networks of increasing sizes up to n=2,048 with $|\mathcal{E}| = 2n$ for each n using Hierarchical network and exponential diffusion model on cascade data containing 10,000 cascades. We also generate 100 extra cascades with 20%-validation and 80%-test. Figure 4 (a)–(b) shows the MAE of influence (Inf) and infection probability (Prob) estimated by NMF versus time for varying n, which indicate that the error remains low for large networks.

Scalability We compare NMF to InfluLearner in terms of runtime for the influence estimation. For InfluLearner, we draw 200 features. For NMF, the batch size of training cascade data is set to 50 for the network with more than 2,048 nodes, and is 100 for smaller networks. The training is terminated when the average MAE of infection probability on validation data does not decrease for 20 epochs. Figure 4 (c) shows the comparison on runtime (in seconds) of training as we increase the network size n in InfluLearner and NMF. Note that the original implementation of InfluLearner [12] is in

Matlab and the computational time increases drastically in network density, whereas our method retains similar runtime regardless of network density.

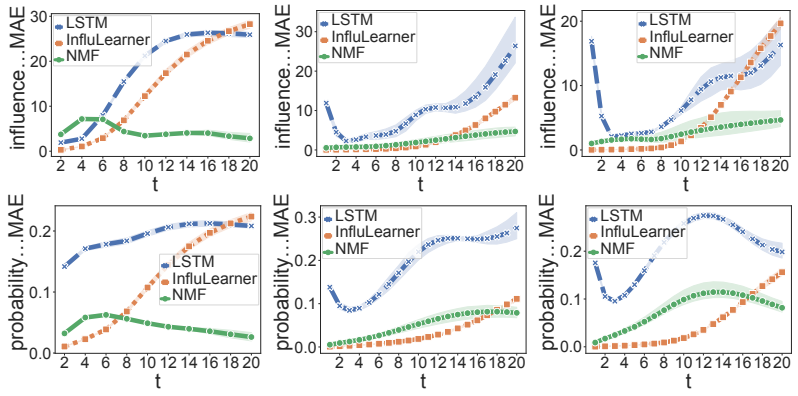


(a) Inf MAE vs t (b) Prob MAE vs t (c) Runtime vs # nodes

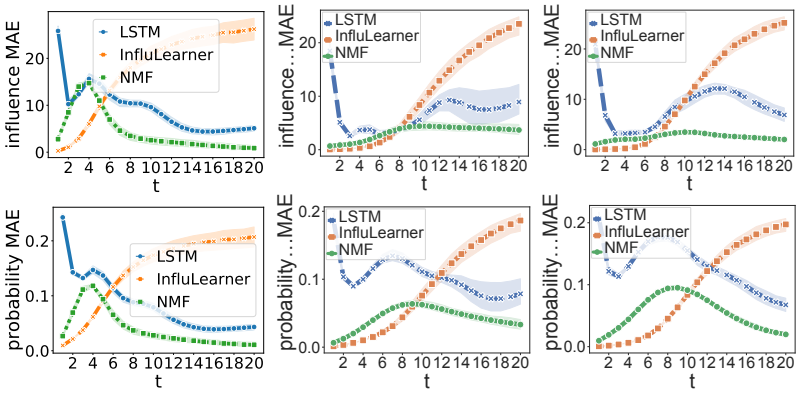
Figure 4: (a)–(b) MAE of influence (Inf) and infection probability (Prob) estimated by NMF for Hierarchical networks with increasing network sizes from 256 to 2048. (c) runtime (in seconds) for training versus network sizes in log-log scale.

D.4 Additional results of infection probability estimation

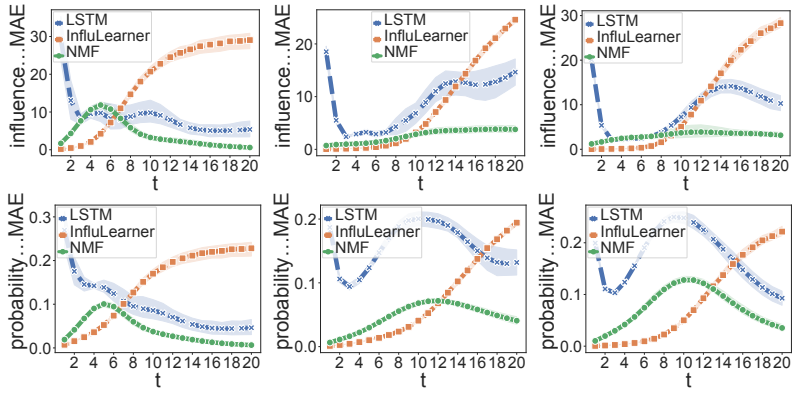
We test a total of 9 combinations of network structures and diffusion models. Specifically, we generate Hierarchical (Hier), Core-periphery (Core), and Random (Rand) networks, and use Exponential (Exp), Rayleigh (Ray) and Weibull (Wbl) diffusion models on each of these networks. All scale and shape parameters are drawn from Unif$[0.1, 1]$ and Unif$[1, 10]$, respectively. Here we stretch NMF and apply to Weibull diffusion model even it has two parameters for each edge. The experiment setting and evaluation metrics are the same as in Section 4. The MAE of influence and node infection probabilities are shown in Figure 5, which shows that NMF consistently performs well with low estimation error after trained by cascade data. Again, it is worth noting that InfluLearner requires the identity of source node for every infection in the entire cascade during training, which is generally not available in practice nor needed in NMF.



(a) Hier + Exp (b) Hier + Ray (c) Hier + Wbl



(d) Core + Exp (e) Core + Ray (f) Core + Wbl



(g) Rand + Exp (h) Rand + Ray (i) Rand + Wbl

Figure 5: MAE of influence (top) and node infection probability (bottom) by LSTM, InfluLearner, and NMF on each of the 9 different combinations of Hierarchical (Hier), Core-periphery (Core) and Random (Rand) networks, and exponential (Exp), Rayleigh (Ray) and Weibull (Wbl) diffusion models. Mean (centerline) and standard deviation (shade) over 100 tests are shown.