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  
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 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
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 prediction 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 $k$th component (or $(i,j)$-th entry). All vectors are column vectors unless otherwise noted. We follow the Matlab syntax and use $[\mathbf{x}; \mathbf{y}]$ to denote the vector that stacks $\mathbf{x}$ and $\mathbf{y}$ vertically. We denote inner product by $\mathbf{x} \cdot \mathbf{y}$ and component-wise multiplication by $\mathbf{x} \odot \mathbf{y}$. 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 $\frac{\partial}{\partial t}$ is with respect to $t$, and gradient $\nabla_{\mathbf{x}}$ is with respect to $\mathbf{x}$. Probability is denoted by $\Pr(\cdot)$, and expectation with respect to $\mathbf{X}$ (or $p_X$) is denoted by $\mathbb{E}_\mathbf{X}[\cdot]$.

**Diffusion network models** Consider a diffusion network model, which consists of a network (directed graph) $G = (V, E)$ with node set $V = [n] := \{1, \ldots, n\}$ and edge set $E \subset V \times V$, and a diffusion model that describes the distribution $p(t; \alpha_{ij})$ of the time $t$ node $i$ takes to infect a healthy neighbor $j \in \{j' : (i, j') \in E\}$ for every $(i, j) \in E$. Then, given a source (seed) set $S$ 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 $S$ 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 $D$ of a diffusion network are often in the form of sample cascades $D := \{C_k = (S_k, \tau_k) \in 2^V \times \mathbb{R}^+_0 : k \in [K]\}$, where the $k$th cascade $C_k$ records its source set $S_k \subset V$ and the time $(\tau_k)_i \geq 0$ which indicates when node $i$ was infected (if it was not infected during $C_k$ then $\tau_k)_i = \infty$). We also equate $C_k$ with $\{\hat{x}^{(k)}(t) \in \{0, 1\}^n : i \in [n], t \geq 0\}$ such that $(\hat{x}^{(k)}(t))_i = 1$ if the node $i$ is in the infected status at time $t$ and 0 otherwise. For example, $\hat{x}^{(k)}(0) = \chi_{S_k}$ where $(\chi_{S_k})_i = 1$ if $i \in S_k$ and 0 otherwise. Such cascade data are collected from historical events for training purposes.

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

$$\mathbf{x}(t; \chi_S) = [x_1(t; \chi_S), \ldots, x_n(t; \chi_S)]^\top \in [0, 1]^n$$

(1)
for all $t \geq 0$ and any source set $S \subset V$. In [1], $x_i(t; \chi_S)$ is the probability of node $i$ being infected at time $t$ given a source set $S$ (not necessarily observed as a source set in $D$). Note that we use $\chi_S$ and $S$ interchangeably hereafter. The probability $x(t; \chi_S)$ can also be used to compute the influence function $\sigma(t; S) := 1_N x(t; \chi_S)$, 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 $E$ and $A$ given cascade data $D$. The matrix $A$ 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 $D$ are available, resulting in a two-stage approach: a network inference is performed first to learn the network structure $E$ and the diffusion model parameters $A$, and then an influence estimation is used to compute the influence for the source set $S$. However, approximation errors and biases in the two stages will certainly accumulate. Alternatively, one can use a one-stage approach to directly estimate $x(t; \chi_S)$ of any $S$ from the cascade data $D$, which is more versatile and less prone to diffusion model misspecification. Our method is a similar kind of one-stage method. Additionally, it allows knowledge of $E$ and/or $A$, if available, to be integrated for further performance improvement.

**Influence maximization** Given cascade data $D$, influence maximization is to find the source set $S$ that generates the maximal influence $\sigma(t; S)$ 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

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

In the next subsection, we consider the special case where $n = 2$. There are two main ingredients of an influence maximization method for solving [2]: an influence prediction subroutine that estimates the influence $\sigma(t; S)$ for any given source set $S$, and an (approximate) combinatorial optimization solver to find the optimal set $S$ of [2] repeatedly called the subroutine. The combinatorial optimization problem is NP-hard and is often approximately solved by greedy algorithms with guaranteed sub-optimality when $\sigma(t; S)$ is submodular in $S$. 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 $\chi_S$, let $X_i(t; \chi_S)$ 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 $X_i(t; \chi_S)$ 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 $X_i(t; \chi_S)$ given the history $H(t) = \{X_i(s; \chi_S) : s \leq t, i \in [n]\}$, i.e.,

$$
\lambda^*_i(t) := \lim_{\tau \to 0^+} \frac{E[X_i(t + \tau; \chi_S) - X_i(t; \chi_S)|H(t)]}{\tau},
$$

(3)

Note that the numerator of (3) is also the conditional probability $\Pr(X_i(t + \tau) = 1, X_i(t) = 0|H(t))$ for any $\tau > 0$. In influence prediction, our goal is to compute the probability $x(t; \chi_S)$ given a source set $S$ conditioning on $H(t)$:

$$
x_i(t; \chi_S) = \mathbb{E}_{H(t)}[X_i(t; \chi_S)|H(t)].
$$

(4)

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

$$
x_{\mathcal{I}}(t) = \mathbb{E}_{H(t)}[\prod_{i \in \mathcal{I}} X_i(t; \chi_S)|H(t)], \quad y_{\mathcal{I}}(t) = \prod_{i \in \mathcal{I}} x_i(t), \quad e_{\mathcal{I}}(t) = x_{\mathcal{I}}(t) - y_{\mathcal{I}}(t)
$$

(5)

for any $\mathcal{I} \subset [n]$ and $|\mathcal{I}| \geq 2$. Then we can derive the evolution of $z := [x; e]$. Here $x(t) \in [0, 1]^n$ is the resolved variable whose value is of interests and samples can be directly observed from the cascade data $D$, and $e(t) = [e_1(t), \ldots] \in \mathbb{R}^{N - n}$ where $N = 2^n - 1$ is the unresolved variable that captures all the second and higher order moments. The complete evolution equation of $z$ is given in the following theorem, where the proof is provided in Appendix B.1.
Theorem 1. The evolution of \( z(t) = [x(t); e(t)] \) follows the nonlinear differential equation:

\[
  z' = \tilde{f}(z), \quad \text{where} \quad \tilde{f}(z) = \tilde{f}(x, e) = \left[ f(x; A) - (A \otimes E)1; \ldots, f_I(x, e); \ldots \right],
\]

with initial value \( z_0 = [x_S; 0] \in \mathbb{R}^N, \quad E = [e_{ij}] \in \mathbb{R}^{n \times n}, \) and

\[
  f(x; A) = Ax - \text{diag}(x)Ax,
\]

\[
  f_I(x, e) = \sum_{i \in I} \sum_{j \notin I} \alpha_{ij}(y_t - y_{I \cup \{j\}} + e_t - e_{I \cup \{j\}}) - \sum_{i \in I} y_{I \setminus \{i\}} \sum_{j \notin I} \alpha_{ij}(x_j - y_{ij} - e_{ij}).
\]

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 \( z \) 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 \( x \) 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 \( x \) part of the dynamics [6]. The GLE of \( x \) is derived from the original equation (6) describing the evolution of \( z = [x; e] \), while maintaining the effect of the unresolved part \( e \). This is particularly useful in our case, as we only need \( x \) for infection probability estimation and influence prediction.

Define the Liouville operator \( \mathcal{L} \) such that \( \mathcal{L}g(z) := \tilde{f}(z) \cdot \nabla_z g(z) \) for any real-valued function \( g \) of \( z \). Let \( e^{t\mathcal{L}} \) be the Koopman operator associated with \( \mathcal{L} \) such that \( e^{t\mathcal{L}}g(z(0)) = g(z(s)) \) where \( z(t) \) solves (6). Then \( \mathcal{L} \) is known to satisfy the semi-group property for all \( g \), i.e., \( e^{t\mathcal{L}}g(z) = g(e^{t\mathcal{L}}z) \). Now consider the projection operator \( \mathcal{P} \) as the truncation such that \( (\mathcal{P}g)(z) = (\mathcal{P}g)([x; e]) = g([x; 0]) \) for any \( z = [x; e] \), and its orthogonal complement as \( Q = I - \mathcal{P} \) where \( I \) is the identity operator.

The following theorem describes the exact evolution of \( x(t) \), and the proof is given in Appendix B.2.

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

\[
  x' = f(x; A) + \int_0^t k(t - s, x(s)) \, ds,
\]

where \( f \) is given in (7), and \( k(t, x) := \mathcal{P}Le^{tQ\mathcal{L}}QAx \).

Note that, (9) is not an approximation—it is an exact representation of the \( x \) 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 \( x \), 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 \( x \), we consider an approximation of the Mori-Zwanzig memory term by a neural net \( e \) with time convolution of \( x \) as follows,

\[
  \int_0^t k(t - s, x(s)) \, ds \approx e(x(t), h(t); \eta) \quad \text{where} \quad h(t) = \int_0^t K(t - s; w) x(s) \, ds.
\]

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

\[
  x' = f(x, h; \theta) := f(x; A) + e(x, h; \eta).
\]

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

\[
  x(0) = x_S, \quad h(0) = 0, \quad \text{and} \quad x(t) = h(t) = 0, \quad \forall t < 0.
\]

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 \( (x, h) \). In addition, we show that the discretization of this system reduces to a structured recurrent neural network if \( K(t; w) \) 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:

\[ x' = f(x, h; A, \eta) = f(x; A) + \varepsilon(x, h; \eta) \]  
\[ h' = \int_0^t K(t-s; w) f(x(s), h(s); A, \eta) \, ds \]

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

\[ x_{t+1} = x_t + f(x_t; A) + \varepsilon(x_t, h_t; \eta) \]  
\[ h_{t+1} = h_t + \sum_{l=1}^L (B_l x_{t+1} - C_l h_t) \]

where the input is given by \( x_0 = \chi_S \) and \( h_0 = 0 \).

The proof is given in Appendix B.3. The matrices \( B_l \) and \( C_l \) in (14b) correspond to the weights on \( x_{t+1} \) and \( h_t \) to form the linear transformation, and the neural network \( \varepsilon \) wraps up \( (x_t, h_t) \) to approximate the nonlinear effect of the memory term in (10).

We here consider a more general convolution kernel \( K(\cdot; w) \) than the exponential kernel. Note that, in practice, the convolution weight \( K \) on older state \( x \) in (10) rapidly diminishes, and hence the memory kernel \( K \) 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

\[ h_t = K^w m_t \quad \text{where} \quad K^w = [K_0^w, \ldots, K_T^w] \quad \text{and} \quad m_t = [x_t; \ldots; x_{t-\tau}] \]

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

\[ m_{t+1} = g(m_t; \theta), \quad \text{where} \quad g(m_t; \theta) := [J_0 m + \hat{f}(J_0 m, K^w m_t; \theta); J_0 m; \ldots; J_{\tau-1} m] \]

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

An optimal control formulation of parameter learning. Now we consider the training of the network parameters \( \theta = (A, \eta, w) \) of (16) using cascade data \( D \). Given a sample cascade \( \hat{x} = (S, \tau) \) from \( D \), 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., \( \hat{x} = \{ \hat{x}_t : t \in [T] \} \) (see Section 2). Maximizing the log-likelihood of \( \hat{x} \) for the dynamics \( x_t = x_t(\theta) \in [0, 1]^n \) induced by \( \theta \) is equivalent to minimizing the loss function \( \ell(\hat{x}, x) \):

\[ \ell(\hat{x}, x) = \sum_{t=1}^T \hat{x}_t \cdot \log x_t + (1 - \hat{x}_t) \cdot \log(1 - x_t), \]

where the logarithm is taken componentwise. We can add a regularization term \( r(\theta) \) to (17) to impose prior knowledge or constraint on \( \theta \). In particular, if \( \hat{E} \) is known, we can enforce a constraint such that \( A \) must be supported on \( \hat{E} \) only. Otherwise, we can add \( \|A\|_1 \) or \( \|A\|_0 \) (the \( l_1 \) or \( l_0 \) norm of the vectorized \( A \)) if \( \hat{E} \) is expected to be sparse. In general, \( A \) can be interpreted as the convolution to be learned from a graph convolution network (GCN) [26,53]. The support and magnitude of \( A \) 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 \( \theta \) 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 \( \theta \) that steers \( m_t \) to fit data \( D \) through the NMF in (16):

\[ \min_{\theta} \quad J(\theta) := \frac{1}{K} \sum_{k=1}^K \ell(x^{(k)}, \hat{x}^{(k)}) + r(\theta) \]  
\[ \text{s.t.} \quad m_{t+1}^{(k)} = g(m_t^{(k)}; \theta), \quad m_0^{(k)} = [\chi_{S_0}, 0, \ldots, 0], \quad t \in [T] - 1, \quad k \in [K], \]

where \( x_t^{(k)} = J_0 m_t^{(k)} \) 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 [14] or [16] require a time invariant control \( \theta \) throughout. This is necessary since \( \theta \) corresponds to the network parameter and needs to be shared across different layers of the RNN, either from the linear kernel case with state \( [x; h] \) in [14] or the general case with state \( m \) 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

\[
H(m, p; \theta) = p \cdot g(m; \theta) - \frac{1}{T} v(\theta),
\]

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

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

\[
m^*_t+1 = g(m^*_t; \theta^*), \quad m^*_0 = [\chi S_0; 0; \ldots; 0], \quad t = 0, \ldots, T-1,
\]

\[
p^*_t = p^*_{t+1} \cdot \nabla_m g(m^*_t; \theta^*), \quad p^*_T = -\nabla \ell, \quad t = T-1, \ldots, 0.
\]

Moreover, the optimal \( \theta^* \) maximizes the total Hamiltonian: for any \( \theta \), there is

\[
\sum_{t=0}^{T-1} H(m^*_t, p^*_{t+1}; \theta^*) \geq \sum_{t=0}^{T-1} H(m^*_t, p^*_{t+1}; \theta).
\]

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

The proof is given in Appendix B.4. We introduced the total Hamiltonian \( \sum_{t=0}^{T-1} H(m_t, p_{t+1}; \theta) \) in Theorem 4 since the NMF dynamics [14] (or [16]) suggest a time invariant control \( \theta \) independent of \( t \), which corresponds to \( \theta \) 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 \( J \) 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 \( D \), NMF can learn the parameter \( \theta = (\mathbf{A}, \eta, w) \). The support (indices of nonzero entries) of the matrix \( \mathbf{A} \) reveals the edge \( E \) of the diffusion network \( G = (V,E) \), and the values of \( \mathbf{A} \) are the corresponding infection rates on the edges. In addition to the diffusion network parameters inferred by \( \mathbf{A} \), we can also estimate (predict) the influence \( \{x^*_t : t \in \mathcal{T}\} \) of any new source set \( x^*_0 \in \mathbb{R}^n \) by a forward pass of NMF dynamics [16] with the learned \( \theta \). 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 [23] are used: hierarchical (Hier) [3] and core-periphery (Core) [23] 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 \( \alpha_{ij} \) 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
1.78 million users and 308 million following relationships among them. Following the setting in 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. The results are shown 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 test robustness of NMF for varying network density $|\mathcal{E}|/n$. The MAE of influence and infection probability by NMF on a hierarchical network with $n = 128$ are shown in Figure 2a 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 $\mathbf{A}$. In this test, we examine the quality of the learned $\mathbf{A}$. We set the recovered adjacency matrix $\hat{\mathcal{E}}$ to the binary indicator matrix $\mathbf{A}^\top \geq A^\top \geq 0$, i.e., $(\mathcal{E})_{i,j} = 1$ if $(\mathbf{A})_{i,j} \geq 0.01$. To evaluate the quality of $\hat{\mathcal{E}}$ and $\mathbf{A}$, we use four metrics: precision (Prec), recall (Rcl), accuracy (Acc), and correlation (Cor), defined as follows,

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

where $\mathcal{E}^*$ and $\mathbf{A}^*$ 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, $\|\mathbf{A}\| = \text{tr}(\mathbf{A}^\top \mathbf{A})$ is the Frobenius norm of the matrix $\mathbf{A}$. Prec 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 $\mathbf{A}$ and $\mathbf{A}^*$ 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 $\mathbf{A}$ with Rayleigh distribution. Evaluation by four metrics are shown in Table 1 which indicates that

![Algorithm 1](https://example.com/algorithm1.png)

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

**Input:** $\mathcal{D} = \{C_k : k \in [K]\}$ where $C_k = \{x^{(k)}(t) \in \{0, 1\}^n : t = 0, 1, \ldots, T\}$.

**Initialization:** Parameter $\theta = (\mathbf{A}, \eta, \omega)$.

**for** $k = 1, \ldots, K$ **do**

- Sample a mini-batch $\bar{\mathcal{D}} \subset \mathcal{D}$ of cascades.
- Compute $\{m_{t} : t \in [T]\}$ using (16) with $\theta$ and $m_{0} = [\chi_{S} : 0]$ for each $C \in \bar{\mathcal{D}}$. (Forward pass)
- Compute $\nabla_{\theta} \mathcal{J} = \sum_{C \in \bar{\mathcal{D}}} \nabla_{\theta} J(x, \hat{x})$ with $\ell$ in (17). (Backward pass)

**end for**

**Output:** Network parameter $\theta$.
Hier + Ray

Core + Exp

probability MAE

influence MAE

/uni00000053/uni00000055/uni00000052/uni00000045/uni00000044/uni00000045/uni0000004c/uni0000004f/uni0000004c/uni00000057/uni0000005c/uni00000003/uni00000030/uni00000024/uni00000028

/uni00000053/uni00000055/uni00000052/uni00000045/uni00000044/uni00000045/uni0000004c/uni0000004f/uni0000004c/uni00000057/uni0000005c/uni00000003/uni00000030/uni00000024/uni00000028

/uni0000004c/uni00000051/uni00000049/uni0000004f/uni00000058/uni00000048/uni00000051/uni00000046/uni00000048/uni00000003/uni00000030/uni00000024/uni00000028

/uni0000004c/uni00000051/uni00000049/uni0000004f/uni00000058/uni00000048/uni00000051/uni00000046/uni00000048/uni00000003/uni00000030/uni00000024/uni00000028

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 \( A \) are measured by precision (Prc), recall (Rcl), accuracy (Acc), and correlation (Cor). Higher value indicates better quality.

<table>
<thead>
<tr>
<th>Network</th>
<th>Method</th>
<th>Prc</th>
<th>Rcl</th>
<th>Acc</th>
<th>Cor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>NETRATE</td>
<td>0.481</td>
<td>0.399</td>
<td>0.434</td>
<td>0.465</td>
</tr>
<tr>
<td></td>
<td>NMF</td>
<td><strong>0.858</strong></td>
<td><strong>0.954</strong></td>
<td><strong>0.903</strong></td>
<td><strong>0.950</strong></td>
</tr>
<tr>
<td>Hierarchical</td>
<td>NETRATE</td>
<td>0.659</td>
<td>0.429</td>
<td>0.519</td>
<td>0.464</td>
</tr>
<tr>
<td></td>
<td>NMF</td>
<td><strong>0.826</strong></td>
<td><strong>0.978</strong></td>
<td><strong>0.893</strong></td>
<td><strong>0.938</strong></td>
</tr>
<tr>
<td>Core-periphery</td>
<td>NETRATE</td>
<td>0.150</td>
<td>0.220</td>
<td>0.178</td>
<td>0.143</td>
</tr>
<tr>
<td></td>
<td>NMF</td>
<td><strong>0.709</strong></td>
<td><strong>0.865</strong></td>
<td><strong>0.779</strong></td>
<td><strong>0.931</strong></td>
</tr>
</tbody>
</table>

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

**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 DIFFCEL[42] for influence maximization (IM). Like NMF+Greedy, DIFFCEL 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 DIFFCEL but not ours) on a hierarchical network of 128 nodes and 512 edges, and use exponential distribution for the transmission function with \( A \) generated from Unif[1,10]. Time window is \( T = 20 \). For each source set size \( n_0 = 1, \ldots, 10 \), NMF+Greedy and DIFFCEL are applied to identify the optimal source sets, whose influence are computed by averaging 10,000 MC simulated cascades. Figure [42] shows that the source sets obtained by NMF+Greedy generates greater influence than DIFFCEL 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
Influ MAE vs density

We proposed a novel framework using neural mean-field dynamics for inference and estimation on (review of related work on network structure inference is provided in Appendix C), such as node views, likes, comments, reposts) and extract more powerful features in [55]. In methods based on approaches in accuracy and efficiency on both synthetic and real-world data sets.

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 [5], 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 discretization. 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 when he was on leave from College of Computing, Georgia Institute of Technology.

References


