Modeling Information Diffusion in Networks with Unobserved Links

Quang Duong

Michael P. Wellman

Satinder Singh

Computer Science and Engineering, University of Michigan
{qduong,wellman,baveja}@umich.edu

Abstract—Modeling information diffusion in networks enables reasoning about the spread of ideas, news, opinion, and technology across a network of agents. Existing models generally assume a given network structure, in practice derived from observations of agent communication or other interactions. In many realistic settings, however, observing all connections is not feasible. We consider the problem of modeling information diffusion when the network is only partially observed, and investigate two approaches. The first learns graphical model potentials for a given network structure, compensating for missing edges through induced correlations among node states. The second attempts to learn the missing connections directly. Using data generated from a cascade model with different network structures, we empirically demonstrate that both methods improve over assuming the given network is fully observed, as well as a previously proposed structure-learning technique. We further find that potential learning outperforms structure learning when given sufficient data.

I. MODELING INFORMATION DIFFUSION

Information diffusion can be seen at work in a wide range of scenarios, such as news propagation, political grass-root campaigning, product promotion, and technology adoption. Models of information diffusion on social networks, for instance, can inform viral marketing campaigns by offering predictions and analyses of product popularity and advertising effectiveness, given information about early buyers’ behaviors [Leskovec et al., 2007]. They can also be employed to assess the count of agents in a given network structure, in practice derived from observations [Kleinberg, 2007], a node or agent is in one of a finite set of states (e.g., having particular information, adopting a technology, etc.) at a given time. In each discrete time period, each agent decides what state to adopt in the next period, as a probabilistic function of the states of its neighbors.

Formally, let \( G \) be a network of \( n \) agents over a discrete time horizon \( T \). We restrict attention here to capturing the diffusion of a single bit of information, so at time \( t \), each agent \( i \) can be in either of two states: \( s_i^t = 1 \) indicates that agent \( i \) has been infected, and \( s_i^t = -1 \) otherwise. An undirected edge \((i, j)\) in \( G \) represents a connection between agents \( i \) and \( j \): conceptually, that \( i \) and \( j \) interact and may be aware of each other’s state. For each node \( i \), \( N_i \) denotes the neighborhood of \( i: N_i = \{ \{ j \mid (i, j) \in G \} \cup \{ i \} \}. \) Let \( s^t = s_{1, \ldots, n} \) be the state of all agents at time \( t \).

In the version of information diffusion we study, infection persists: there exists no agent \( i \) and time \( t \) such that \( s_i^t = 1 \) and \( s_i^{t+1} = -1 \). We define a binary feature \( a(s_i^{t}, s_i^{t-1}) = 1 \) to indicate that \( i \) becomes infected at time \( t \) (\( s_i^{t-1} = -1 \) and \( s_i^t = 1 \)), otherwise \( a(s_i^{t}, s_i^{t-1}) = -1 \). We define \( t^* \) as the time when \( i \) becomes infected, and \( c(s_{N_i}^t) = |\{ j \in N_i \mid s_j^t = 1 \}| \), the count of agents in \( N_i \) who are infected at \( t \). Similarly, let \( \sigma(s_{N_i}^t, s_{N_i}^{t-1}) = |\{ j \in N_i \mid a(s_j^t, s_j^{t-1}) = 1 \}| \), the number of agents in \( N_i \) that become infected at time \( t \).

The most popular model of infection is the cascade model [Kempe et al., 2003], in which the tendency to infect increases with the proportion of infected neighbors. Goldberg et al. [2001] proposed a cascade model form:

\[
\Pr(s_i^t = 1 \mid s_{i}^{t-1} = 1, s_{N_i-\{i\}}^{t-1}) = 1
\]

\[
\Pr(s_i^t = 1 \mid s_{i}^{t-1} = -1, s_{N_i-\{i\}}^{t-1}) = 1 - (1 - \alpha)(1 - \beta)^{c(s_{N_i}^{t-1})}.
\]

In this model, \( \beta \in [0, 1] \) represents the tendency of infection from interacting with one of its infected neighbors, and \( \alpha \in [0, 1] \) reflects the possibility of node \( i \) getting information from sources other than its neighbors, or in other words spontaneously becoming infected.\(^1\) For example, when \( \alpha = 0, \)

\(^1\)The cascade model entails that influence is uniform across a node’s neighbors and all network nodes will eventually become infected. We adopt this model not because we necessarily desire these characteristics, but because the model is widely applied in the literature. Nothing about our approach is particularly geared to these properties.
\[ \beta > 0, \text{ and } c(s^{t-1}_{N_i}) = 1, \text{ } i \text{ does not get information from sources other than its neighbors, and can become infected only from interacting with its sole infected neighbor with probability } 1 - (1 - \beta) = \beta. \]

Stonedahl et al. [2010] introduced an alternative cascade model, which we label \( C \), that induces similar behavior based on the same intuitions, but with a simplified probability expression:

\[
\Pr(s^t_i = 1 | s^{t-1}_{N_i}) = \alpha + \beta \frac{c(s^{t-1}_{N_i})}{|N_i|}. \tag{2}
\]

Assuming independence of agent states given past states, the cascade models define a joint distribution of states at time \( t \):

\[
\Pr(s^t | s^{t-1}) = \prod_i \Pr(s^t_i | s^{t-1}_{N_i}). \tag{3}
\]

Although the cascade models allow for positive probability of infection even if no known neighbors are infected, its accuracy suffers if the network structure is missing connections. Gomez-Rodriguez et al. [2010] were first to identify and tackle the problem of discovering underlying network structure given only diffusion history. They introduce an algorithm called NetInf, which identifies a network that optimizes an approximate measure of fit to observed infection times.\(^2\) In the present work, we consider the structure learning approach: a greedy algorithm of our own, named MaxLinInf, as well as a version NetInf\(^\prime\) of the previous algorithm modified to fit our problem setup. We further introduce a fundamentally different approach, potential learning, which can compensate for missing edges by capturing induced correlations of behavior in potential functions that encompass nodes not directly connected in the given graph.

II. PROBLEM SETUP

Let \( G \) be the observed network, and \( S = \{s\} \) a set of diffusion instances or traces, each of length \( T \), \( s = \{s^0, \ldots, s^T\} \). We assume that the diffusion was generated by some process (in our empirical study, the cascade model) on a true underlying network \( G^* \supseteq G \). Given \( G \) and \( S \), we would like to make inferences about future diffusion events. We next provide an example of the problem and formally define our objectives.

A. Example

Consider the example four-node scenario shown in Figure 1, where the edge between nodes 1 and 3 is not observed. Suppose that in the true underlying model nodes can become infected only from interacting with their neighbors. In the example run, the spread started with node 1 at time \( t = 1 \) and reached nodes 4 and 3 at \( t = 5 \) and \( t = 6 \), respectively. The missing edge \((1, 3)\) may mislead us to interpret node 3’s infection as spontaneous. Throughout Section III, we provide more details about how each approach in this study compensates for the missing connection \((1, 3)\) in this particular example.

\(^2\)Gomez-Rodriguez et al. [2011] introduced an improved version of NetInf that treats the underlying diffusion as a continuous-time process and learns the model parameters from data.

B. Evaluation

We capture the dynamics of information diffusion using the joint probability distribution of agent states, conditional on past states. Notationally, \( \Pr^M(s^t | s^{t-1}) \) is the joint distribution of agent states at time \( t \) induced by a particular model \( M \). Given a set \( S \) of \( m \) traces, we can compute the data’s average logarithmic likelihood induced by \( M \),

\[
L^M(S) = \frac{1}{m} \sum_{s \in S} \frac{1}{T} \sum_t \log \Pr^M(s^t | s^{t-1}). \tag{4}
\]

As in many information network studies, we are also interested in predicting the aggregate extent of future network infection, given some initial diffusion data. In particular, given an initial network state \( s^0 \), we sample diffusion traces of length \( T \) from the model \( M \) starting with \( s^0 \). These traces define a distribution over the fraction \( c(s^T)/n \) of nodes that are infected at time \( T \), denoted as \( Q^M(c(s^T)/n | s^0) \). We employ a version of the Kullback-Leibler (KL) divergence metric, the skew divergence [Lee, 1999], to measure the distance between the two discrete value distributions \( Q \) induced by the true model \( M^* \) and by the test model \( M \), conditional on observations of the initial state \( s^0 \). We specify the KL divergence as:

\[
d_{KL}(Q^M, Q^M | s^0) = \sum_{c=0}^{n} Q^M(c/n | s^0) \log \frac{Q^M(c/n | s^0)}{Q^M(c/n | s^0)},
\]

and average skew divergence given data \( S \),

\[
D(Q^M, Q^M | S) = \frac{1}{m} \sum_{s^0 \in S} d_{KL}(Q^M, \rho Q^M + (1 - \rho)Q^M | s^0),
\]

with \( \rho \) set at 0.99 to avoid the problem of undefined KL divergence values [Lee, 1999].

We can further measure the quality of the graphs learned by MaxLinInf and NetInf\(^\prime\) in simulated scenarios where access to the true underlying graph is available. In particular, we employ the following metrics on structural differences between the true underlying graph \( G^* = (V, E^*) \) and the learned structure \( G' = (V, E') \):\(^3\)

\[
\delta_-(G^*, G') = \frac{|E^* \setminus E'|}{|E^*|}, \quad \text{and} \quad \delta_+(G^*, G') = \frac{|E' \setminus E^*|}{|E'|}.
\]

\(^3\)A version of these metrics was also employed in evaluating the original NetInf [Gomez-Rodriguez et al., 2010].

}\(t=5\) \[4\] \[2\] \[3\] \[4\] \[2\] \[3\] \[1\] \[3\] \[1\] \[3\]
Here $\delta_-(G^*, G')$ represents the proportion of false negative edges in the true network $G^*$, or in other words, edges that remain missing in the learned graph $G'$. Similarly, $\delta_+(G^*, G')$ captures the proportion of false positive edges in the learned structure $G'$.

### III. Dealing with Partially Observed Networks

We consider two broad approaches to learning diffusion models despite unobserved network edges. The potential learning approach captures probabilistic dependencies induced by the missing edges in the potential function of a graphical model. The structure learning approach attempts to recover the edges explicitly by learning network structure.

#### A. Graphical Multiagent Models

A graphical multiagent model (GMM) is a graphical model with vertices $V$ representing the $n$ agents, and edges $E$ capturing pairwise interactions between them [Duong et al., 2008]. We employ here a dynamic extension, the history-dependent graphical multiagent model (hGMM) [Duong et al., 2010], which captures agent interactions in dynamic scenarios by conditioning joint states on abstracted history. We limit conditioning to the previous state, associating with each agent $i$ a potential function $\pi_i(s_{Ni} \mid s_{Ni}^{-1})$. The potential of a local state configuration $s_{Ni}$ represents its unnormalized likelihood of being included in the global outcome, conditional on past state [Kakade et al., 2003, Daskalakis and Papadimitriou, 2006, Duong et al., 2010]:

$$\Pr(s_i \mid s_{i-1}) = \frac{1}{Z} \prod_i \pi_i(s_{Ni} \mid s_{Ni}^{-1}).$$

Unlike the cascade models (3), hGMMs do not assume conditional independence of agent states given history, but specify the joint state directly. That hGMMs compute the potentials of all state configurations of a neighborhood allows reasoning about state correlations between neighbors who appear disconnected in the input graphical structure, and thus helps to compensate for missing edges. We approximate the normalization factor $Z$ using the belief propagation method [Yedidia et al., 2000], which has shown good results with reasonable time in sparse cyclic graphs. In particular, we employ the libDAI approximate inference package [Mooij, 2010].

Figure 2 illustrates an example four-agent hGMM of the scenario described in Section II-A. Note that the figure distinguishes two kinds of edges. Undirected edges define the neighborhoods $N_i^u$ within a time slice, capturing correlations among the nodes in $N_i^u$ at a particular time $t$. Directed edges ending at $i$ capture how past states of the neighborhood $N_i^d$ influence $i$’s present state. This distinction supports a generalized form of potential function, $\pi_i(s_{Ni}^u \mid s_{Ni}^{u-1})$, by allowing that the conditioning nodes be different from the nodes interdependent within a time slice. In contrast, a cascade model of the same scenario would omit the undirected edges, expressing the cascade functional form $\Pr(s_i \mid s_{i-1})$. For most of this paper, we similarly assume that $N_i^u, N_i^d$, and the given $N_i$ are the same for all $i$, and employ the potential function form $\pi_i(s_{Ni}^u \mid s_{Ni}^{u-1})$.

Given data from the example scenario of Section II-A, the cascade models would learn a high value for $\alpha$, assuming spontaneous infections, even though in the true model, information spreads only from node to node. In contrast, hGMMs could use the potential function of node 2 to express correlations between nodes 1 and 3 to compensate for the missing edge (1, 3). For instance, consider neighborhood $N_2 = (1, 2, 3)$ with history $s_{N_2} = (s_1^5, s_2^5, s_3^5) = (1, -1, -1)$, as depicted in Figure 1. Since the potentials $\pi(s_{N_2} = (1, -1, -1) \mid s_{N_2}^5)$ and $\pi(s_{N_2} = (1, -1, 1) \mid s_{N_2}^5)$ govern how node 1 infects node 3, assigning positive values to these potentials enables hGMMs to capture the interaction. Unlike $\pi(s_{N_2} \mid s_{N_2}^5)$, the corresponding cascade-model construct $\Pr(s_3^5 \mid s_{N_2}^5)$ cannot compensate in this way because it inherently assumes independence among nodes in $N_2$.

We introduce hGMMs for information diffusion in two different forms.

1) **Tabular hGMMs:** The tabular hGMM $\text{tabG}$ explicitly specifies the potential function of each neighborhood, $\pi(s_{Ni}^d \mid s_{Ni}^{d-1})$, as a function of five features:

$$c(s_{Ni}^{d-1}), \sigma(s_{Ni}^d, s_{Ni}^{d-1}), \vert N_i \vert, s_{Ni}^{l-1}, s_i^l.$$  

For learning such forms, we treat the potential value corresponding to each configuration of these features as a parameter. The number of parameters thus grows polynomially with the largest neighborhood’s size.

2) **Parametric hGMMs:** The parametric hGMM $\text{paG}$ employs a functional form for potentials based on the cascade model formula (2). We define $g(s_y = 1 \mid s_y)$ as a probability measure of any uninfected node $y$ in the set of nodes $Y$ becoming infected given the current state $s_y$: $g(s_y = 1 \mid s_y) = \alpha + \beta c(s_y)/|Y|$. We overload $g(s_y) = g(s_y = 1 \mid s_y)$, as $g$ is identical for all $y$. Let $N_i' = N_i \setminus \{i\}$ and $c = \sigma(s_{Ni}', s_{Ni}^{l-1})$. The potential function of neighborhood $N_i$ is the product of three terms:

$$\pi(s_{Ni}^d \mid s_{Ni}^{d-1}) = \Pr(s_i^l \mid s_{Ni}^{l-1}; \alpha, \beta)g(s_{Ni}'^{l-1}; \alpha_1, \beta_1)^{\lfloor c - \gamma \vert N_i' \rfloor} \left(1 - g(s_{Ni}'^{l-1}; \alpha_1, \beta_1)\right)^{\lfloor N_i' \rfloor - c}. \quad (5)$$

The first term of (5) represents the probability of node $i$’s infection, given by the cascade model (2). The second term corresponds to nodes in $N_i'$ that become infected at $t$, whereas the final term corresponds to nodes in $N_i'$ not infected at $t$. If we assumed independence of agent states, the exponent in the second term would simply be $c$, as the joint probability of $c$ nodes from $Y$ becoming infected is $g(s_Y)^c$. From our study of $\text{tabG}$, we find that the distance of this count from a fraction of neighborhood size provides a good way to capture correlation among neighbor infections, hence the exponent in the second term $\lfloor c - \gamma \vert N_i' \rfloor$ with $\gamma \in [0, \infty)$. Here $\gamma$ indicates the degree of correlation among agent states. Thus, complete independence of agent states results in $\gamma = 0$. Overall, the
paG model employs seven parameters: $\alpha$, $\beta$, $\alpha_1$, $\beta_1$, $\alpha_{-1}$, $\beta_{-1}$, and $\gamma$.

Our graphical multiagent model approach does not explicitly seek to discover missing edges, but instead takes advantage of the GMM’s flexibility in capturing joint states to model information diffusion on the original network.

B. Learning Graphical Structures

In the structure-learning approach, we attempt to discover unobserved connections using diffusion observation data. We consider two algorithms: one adapted from prior work under a different model, and the second a straightforward greedy learner introduced here.

1) NetInf and NetInf’ Algorithms: Gomez-Rodriguez et al. [2010] employed the independent cascade model to capture information diffusion, and further assumed that every node becomes infected from exactly one other infected neighbor node. Given a graph structure $G$, let $T$ be the set of all possible directed trees $\tau$ whose edges $E_{\tau}$ are a subset of $E_G$. A model $M$ over $G$ computes the likelihood of observing the transmissions in a diffusion instance $s$ on each tree $\tau \in T$ as follows. We first describe the computation of $\Pr(i; s, \tau)$: the probability of observing node $i$ becoming infected in $s$ on tree $\tau$. For each node $i$ such that there (uniquely) exists an edge $(j, i) \in \tau$, $\Pr(i; s, \tau)$ is defined as the probability of transmission from node $j$, infected at time $t_j^*$, to node $i$ at time $t_i^*$, which is a function of $t_i^* - t_j^*$. NetInf employs the power-law and exponential waiting time models in specifying $\Pr(i; s, \tau)$, which would put it at a disadvantage when given data from this study’s generative cascade model. Therefore, the modified version NetInf’ replaces their definition with $\Pr(i; s, \tau) = \beta(1 - \beta)(t_i^* - t_j^*)^{-1}$ if $t_i^* < t_j^*$, and $\Pr(i; s, \tau) = 0$ otherwise. For each $i$ that does not have an incoming edge in $\tau$, the probability of observing node $i$’s spontaneous infection is $\Pr(i; s, \tau) = \alpha$. Model $M$ then interprets the probability of observing $s$ on tree $\tau$: $\Pr(s; \tau) = \prod_{i \in \tau} \Pr(i; s, \tau)$. From this the algorithm approximates the probability of observing $s$ on graph $G$ as that of observing $s$ on the maximum-likelihood tree $\tau_s$: $\Pr(s | G) \approx \max_{\tau \in T} \Pr(s; \tau)$.

Starting with some initial graph $G$, at each step, the NetInf’ algorithm, as well as NetInf, adds an edge $(i, j)$ that maximizes the log likelihood of data $S$: $L_{\text{tree}}(S | G \cup (i, j)) = \sum_{s \in S} \log(\Pr(s | G \cup (i, j)))$. As $L_{\text{tree}}$ is monotone in graph size, this method would produce the complete graph without a limit $K$ on the number of edges that can be added. Note that $\alpha$ and $\beta$ here have the same interpretation as in the cascade model (1). Since NetInf’ does not learn parameters $\alpha$ and $\beta$, we provide it with the parameter values used to generate the input data. We refer to the model learned by NetInf’ as netC.

2) MaxLInf Algorithm: We introduce a similar greedy algorithm MaxLInf, described in pseudocode below. The result $\text{maxC}$ is a cascade model (2). Unlike NetInf’, the MaxLInf algorithm learns the cascade model parameters $\alpha$ and $\beta$ as well as the graphical structure $G'$. The algorithm employs as its objective function the average log likelihood (4) of diffusion instances $S$.

\begin{algorithm}
1: Input $(G, S)$
2: $G' \leftarrow G$
3: Learn $(\alpha, \beta)$ that maximizes $L^{\text{maxC}}(S | G')$
4: repeat
5: Find $(i, j) \notin G'$ maximizing $L^{\text{maxC}}(S | G' \cup (i, j))$
6: if $\Delta(i, j) = L^{\text{maxC}}(S | G' \cup (i, j)) - L^{\text{maxC}}(S | G') > 0$ then
7: $G' \leftarrow G' \cup (i, j)$
8: Learn $(\alpha, \beta)$ that maximizes $L^{\text{maxC}}(S | G')$
9: end if
10: until $\Delta(i, j) > 0$ or $\exists (i, j) \notin G'$
\end{algorithm}

Note that MaxLInf does not require any predetermined constraint on the model’s complexity, since adding more edges does not necessarily increase the likelihood $L$ of input diffusion observations.

3) Discussion: In the example of Section II-A, these two learning algorithms may be able to discover the hidden edge $(1, 3)$, given sufficient observations. Since they are greedy,
however, they may also add spurious edges that distort their 
views of the true network. For example, given some limited 
data set in which the case where node 4 was infected before 
node 3 occurs disproportionally frequently, they may greedily 
add (3, 4) first to best explain the data, and stop before adding 
(1, 3).

IV. Empirical Study

Our empirical study employs graphs of modest size to focus 
on investigating the relative ability of structure learning and 
graphical models to deal with unobserved connections. We 
empirically evaluate models $\text{paG}$, $\text{tabG}$, and $\text{maxC}$ against 
baseline models $\text{C}$ and $\text{netC}$ on simulated data generated 
from the true cascade $G^*$, with various graphical structures 
and experiment settings. We examine their detailed prediction 
performance based on the log likelihood $L$, aggregate pre-
diction performance captured in the skew divergence $D$, and 
graph difference measures $\delta_-$ and $\delta_+$. 

A. Data Generation

Our empirical study consists of two experiment groups: 
(A) stylized graphs of three to five nodes, and (B) larger graphs 
of size 30 and 100. In experiment group B, we employ the 
Erdos-Renyi model (ER) and the Barabasi-Albert preferential 
attachment model (PA) to generate random graphs $G^*$ [Erdos 
and Renyi, 1959, Barabási and Albert, 1999]. The ER model 
randomly creates an edge between each pair of nodes with 
equal probability, independently of the other edges. The PA 
model constructs graphs such that the more connected nodes 
are more likely to receive new connections. We generate data 
from the $G^*$ model on the underlying graphs $G^*$ with time 
horizon $T = 30$, using the $\alpha$ and $\beta$ values that Stonedahl et al. 
[2010] learned from various real-world social networks at two 
different diffusion speeds: normal and fast (viral).

We uniformly randomly remove 50% of the edges from 
each $G^*$ to generate the observable graph $G$, supplied as 
input to all the methods tested. Whereas Gomez-Rodriguez 
et al. [2010] developed and evaluated their original $\text{NetInf}$ algorithm without graph input, we seed the modified version $\text{NetInf}'$ with the partial structure $G$.

We employ the log-likelihood function (4) for training models 
in the methods studied here, except for $\text{NetInf}'$. To learn model parameters, we search for settings that maximize the objective function using gradient descent with early stopping and random restarts.

B. Experiment Settings

Each result data point is averaged over 10 trials. The 
group A trials use 200 diffusion instances for training and 
500 for testing. In experiment group B, we vary the number 
of diffusion traces for training from 25 to 100, and use 200 
instances for testing in each trial. For clarity, we present the 
negated log likelihood measure $-L^M(S)$ for a model $M$; 
lower values indicate better performance. To assess aggregate 
infected statistics for model $M$, we generate 200 diffusion 
traces of length $\hat{T} = 5$ from the initial state of each given 
test instance $s$. From these we construct the conditional dis-
bution $Q^M(c(s^T)/s^0)$. Given the distributions $Q^M$, we 
calculate the aggregate prediction performance for model $M$, 
$D(Q^G, Q^M | S)$. To establish statistical comparisons of two 
models $M_1$ and $M_2$, we further perform bootstrapped paired $t$-
tests for both measures $L$ and $D$. We distinguish three different 
outcomes in comparison tables: $M_1$ outperforms $M_2$ with 
$p < 0.05$ (black), $M_2$ outperforms $M_1$ with $p < 0.05$ (white), 
and neither (gray). We investigate two variants of $\text{NetInf}'$, 
set to induce graphs with 50% (model $\text{netC-small}$) and 100% 
(model $\text{netC-large}$) more edges than the given graph $G$. In 
addition to trials with partially observed networks (partial), 
we include some fully observed cases (full) for calibration 
and sanity checking.

We label our experiments to indicate the following features: 
(i) coverage of the underlying graph by the given graph (partial 
or full), (ii) graph family (PA or ER), (iii) graph size, (iv) fast 
cascade speed in the generative model, and (v) number of input 
diffusion traces (in parenthesis).

C. Results

We present results on detailed prediction performance, 
measured by $L$, for the 5-node case in Figure 3. We used 
the insights gained from analyzing the potential function 
parameters for $\text{tabG}$ in the three- and four-node cases to 
construct $\text{paG}$, but do not report results from these cases 
here. We also omit comparisons of $D$, $\delta_-$, and $\delta_+$, which 
provide little differentiation among the models for such small 
networks.

![Fig. 3. Detailed prediction performance for the 5-node graph (normal and fast spreading speeds).](image)

Models $\text{maxC}$, $\text{tabG}$, and $\text{paG}$ provide better detailed 
prediction performance than the baseline models $\text{C}$ and $\text{netC}$, 
when the input graph has missing edges. When the input graph 
is fully observed, $\text{C}$ is the correct model, and as expected pro-
vides the best predictions. In addition, the hGMMs outperform $\text{maxC}$ in the normal-spreading case but trail $\text{maxC}$ in the 
fast-spreading scenario. In fact, we observe that $\text{maxC}$ con-
tains more correct edges (false negative $\delta_-(G^*, G') = 0.1$) in
the fast spreading case than in the normal case ($\delta_-(G^*, G^f) = 0.25$). As there are more spontaneous infections and fewer inter-node transmissions in the normal case, MaxLInf may too quickly add edges that help to explain spontaneous infections even though these edges are not present in the underlying graph. The statistical test results in Table I further cement confidence in the model comparisons above.

We do not show most of the results for NetInf′, as the models produced by this algorithm perform relatively poorly by our measures, despite the advantage of being given the true cascade model’s parameter values. This discrepancy is likely a consequence of NetInf′’s use of an objective function $L_{t\text{tree}}$ at variance from our use of $L^M(S)$ for evaluation. The respective objective functions are applicable for two different types of network influence data: ordinal-time and snapshot [Cosley et al., 2010]. The objective $L_{t\text{tree}}$ is suitable for ordinal-time data sets, which record the time of each individual node’s change of state. The majority of data sets, however, store snapshots of the network state at various time periods, which align more directly with the objective $L^M(S)$. Future work could investigate the connection between these two objective functions, based on recent work exploring the correspondence between ordinal time and snapshot data [Cosley et al., 2010].

Even for ordinal data, $L_{t\text{tree}}$ is only an approximate measure of the likelihood of observed infection times, and thus does not exactly capture how information diffuses through connections. NetInf′ also requires a predetermined constraint on the final graph’s complexity. With respect to graph difference metrics, NetInf′ is able to discover more missing edges than MaxLInf, resulting in a lower proportion of false negatives $\delta_-$ as depicted in Figure 4. However, NetInf′ also adds significantly more spurious edges than MaxLInf, as reflected by $\delta_+$. This problem worsens as NetInf′ has more freedom to add new edges: NetC-large contains many more spurious edges than NetC-small, although NetC-large is basically informed with the exact number of unobserved connections. Moreover, whereas maxC can compensate for the learned graph’s inaccuracy by fitting its parameters to data, NetC has to use the fixed parameters from the generative cascade model, since the objective function $L_{t\text{tree}}$ is maximized when $\alpha = \infty$ and $\beta = 1$.

Learned parameters of paG can provide additional insights about the underlying network. In particular, analyzing paG’s $\beta_1$ and $\beta_-$ may help to detect if the given network has unobserved edges. Intuitively, given a fully observed network, paG should use $\beta$ solely in capturing information transmissions facilitated by the given network, and set $\beta_1$ and $\beta_-$ to near zero. When there exist missing edges, paG may assign higher values to $\beta_1$ and $\beta_-$ to capture how information diffuses among its seemingly unconnected neighbors. Let us define $b_1 = \beta_1/\beta$ and $b_- = \beta_-/\beta$. Table III shows that both $b_1$ and $b_-$ are notably higher in partially observed graphs than in fully observed structures, confirming this intuition. However, a more thorough examination using a wide range of measures is needed in order to draw a sufficiently confident connection between learned paG parameters and network structure.
Fig. 4. Detailed predictions (left) and aggregate predictions (right) in experiment group B for ER and PA 30-node graphs with different amounts of training data (numbers in parenthesis).

TABLE II
PAIRWISE COMPARISONS OF DETAILED PREDICTIONS (LEFT) AND AGGREGATE PREDICTIONS (RIGHT) FOR EXPERIMENT GROUP B FOR ER AND PA GRAPHS OF 30 AND 100 NODES WITH DIFFERENT AMOUNTS OF TRAINING DATA (NUMBERS IN PARENTHESIS). FOR $M_1$ VS $M_2$ BLACK INDICATES $M_1$ OUTPERFORMS $M_2$ WITH $p < 0.05$, WHITE THE REVERSE, AND GRAY NO SIGNIFICANT DIFFERENCE FOUND.

<table>
<thead>
<tr>
<th>Detailed Predictions</th>
<th>maxC vs C</th>
<th>paG vs C</th>
<th>paG vs maxC</th>
</tr>
</thead>
<tbody>
<tr>
<td>partial-ER-30(25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-ER-30(100)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-ER-30-fast(25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-ER-30-fast(100)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-ER-100(25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-PA-30(25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-PA-30(100)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Aggregate Predictions</th>
<th>maxC vs C</th>
<th>paG vs C</th>
<th>paG vs maxC</th>
</tr>
</thead>
<tbody>
<tr>
<td>partial-ER-30(25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-ER-30(100)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-ER-30-fast(25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-ER-30-fast(100)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-ER-100(25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-PA-30(25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial-PA-30(100)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE III
PARAMETER COMPARISONS IN paG IN EXPERIMENT GROUPS A AND B (DATA AMOUNTS IN PARENTHESES).

<table>
<thead>
<tr>
<th>$b_1$ partial</th>
<th>5 (200)</th>
<th>5-fast (200)</th>
<th>ER (25)</th>
<th>ER-fast (25)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$ full</td>
<td>0.051</td>
<td>0.038</td>
<td>0.24</td>
<td>0.01</td>
</tr>
<tr>
<td>$b_{−1}$ partial</td>
<td>3.87</td>
<td>3.15</td>
<td>2.68</td>
<td>1.722</td>
</tr>
<tr>
<td>$b_{−1}$ full</td>
<td>0.02</td>
<td>0.01</td>
<td>0.03</td>
<td>0.015</td>
</tr>
</tbody>
</table>

V. DISCUSSION
We introduce two solutions to the problem of modeling information diffusion in networks with unobserved edges: learning an hGMM on the given network structure (potential learning), and directly discovering the missing connections (structure learning). We show that our approaches can improve prediction over existing methods in various settings of information diffusion with a considerable number of missing edges. With limited data, our structure-learning algorithm MaxLInf performs best in most scenarios, except for preferential attachment graphs. The hGMM approach produces superior predictions as more data is available, while dominating the other methods in experiments with PA structures. If structure learning could recover the true network reliably, that would clearly be preferred under our assumption that the actual data is generated by the cascade model. The inevitable imperfection of structure learning, however, opens the door to alternative model forms. By relaxing the conditional independence structure of the cascade model, our hGMMs can capture dependencies manifest in the data due to unobserved edges.

NetInf was originally designed for a different evaluation measure, which accounts for its relatively poor showing in our study. It remains possible that some ideas of that method could be incorporated in an algorithm that learns network structures with high predictive accuracy. Future work should explore that possibility as well as other approaches to improve MaxLInf, such as more effective interleaving of structure and parameter learning. Another promising approach is to combine structure learning with graphical models, capturing some unobserved
edges explicitly and some implicitly.

The inference complexity of paG grows exponentially with respect to the largest neighborhood’s size. Although this result enables paG to model medium-size systems, such as academic co-authorship networks [Leskovec et al., 2009] and dynamic consensus experiments [Kearns et al., 2009], handling large social networks on the order of Facebook or Twitter remains a challenge for future work. In one potential approach, we can exploit the potential function’s generalized form, \( \pi_i(s_{N_i}^t | s_{N_i}^t) \), where the neighborhood size of \( s_{N_i}^t \) dictates the inference complexity, while the role of \( s_{N_i}^t \) in the model’s complexity is negligible. In other words, we may choose to include only nodes whose states are most likely to be correlated in the within-time neighborhood \( N_i(t) \), while keeping the conditioning neighborhood \( N_i(t-1) \) intact. Another future direction is to apply the mean field method for approximate inference in graphical models, which in general generates less accurate approximations than the currently employed method, belief propagation [Weiss, 2001]. However, the recent success of the mean field method in comparable settings with large social networks containing tens of thousands of users [Shi et al., 2009] suggests that its employment in GMMs can potentially provide reasonably accurate results and acceptable scalability in large domains.

Acknowledgments

This work was supported in part by the ARL Collaborative Technology Alliance on Network Science.

REFERENCES


