Submodular Inference of Diffusion Networks from Multiple Trees

Manuel Gomez-Rodriguez[1,2]   MANUELGR@STANFORD.EDU
Bernhard Schölkopf[1]   BS@TUEBINGEN.MPG.DE
[1]MPI for Intelligent Systems and [2]Stanford University

Abstract

Diffusion and propagation of information, influence and diseases take place over increasingly larger networks. We observe when a node copies information, makes a decision or becomes infected but networks are often hidden or unobserved. Since networks are highly dynamic, changing and growing rapidly, we only observe a relatively small set of cascades before a network changes significantly. Scalable network inference based on a small cascade set is then necessary for understanding the rapidly evolving dynamics that govern diffusion. In this article, we develop a scalable approximation algorithm with provable near-optimal performance based on submodular maximization which achieves a high accuracy in such scenario, solving an open problem first introduced by Gomez-Rodriguez et al. (2010). Experiments on synthetic and real diffusion data show that our algorithm in practice achieves an optimal trade-off between accuracy and running time.

1. Introduction

Over the last years, there has been an increasing interest in understanding diffusion and propagation processes in a broad range of domains: information propagation (Gomez-Rodriguez et al., 2010), social networks (Kempe et al., 2003), viral marketing (Watts & Dodds, 2007), epidemiology (Wallinga & Teunis, 2004), and human travels (Brockmann et al., 2006).

In the context of diffusion networks, one of the fundamental research problems is how to infer the connectivity of a network based on diffusion traces (Gomez-Rodriguez et al., 2010; 2011; Myers & Leskovec, 2010; Snowsill et al., 2011). In information propagation, we note when a blog or

news site writes about a piece of information. However, in many cases, the blogger or journalist does not link to her source and therefore we do not know where she gathered the information from. In viral marketing, we get to know when customers buy products or subscribe to services, but typically cannot observe the *trendsetters* who influenced customers' decisions. Finally, in epidemiology, we can observe when a person gets ill but cannot tell who infected her. In all these scenarios, we observe spatiotemporal traces of information spread (be it in the form of a meme, a decision, or a virus) but we do not know the paths over which information propagates. We note *where and when* information emerges but not *how or why* it does. In this context, inferring the connectivity of diffusion networks is essential to reconstruct and predict the paths over which information spreads, maximize sales of a product or stop infections.

Our approach to network inference. We consider that on a fixed hypothetical network, diffusion processes propagate as directed trees through the network. Since we only observe the times *when* nodes are reached by a diffusion process, there are many possible propagation trees that explain a set of cascades. Naive computation of the model takes exponential time since there is a combinatorially large number of propagation trees. It has been shown that computations over this super-exponential set of trees can be performed in cubic time (Gomez-Rodriguez et al., 2010). However, to the best of our knowledge, efficient optimization of the model has been an open question to date. Here, we show that computation over the super-exponential set of trees can indeed be performed in quadratic time and surprisingly, we show that the resulting objective function is submodular. Exploiting this natural diminishing property, we can efficiently optimize the objective function to find a near-optimal network with provable guarantees that best explain the observed cascades. Lazy evaluation and the local structure of the problem can be used to speed-up our method. Considering all possible propagation trees enables us to learn a network from fewer observed cascades. This is important since social networks are highly dynamic (Backstrom & Leskovec, 2011), changing and growing rapidly,

and we can only expect to record a small number of cascades over a fixed network.

Related work. The work most closely related to ours (Gomez-Rodriguez et al., 2010; 2011; Myers & Leskovec, 2010) also uses a generative probabilistic model for inferring diffusion networks. NETINF (Gomez-Rodriguez et al., 2010) infers the network connectivity using submodular optimization by considering only the most probable directed tree supported by each cascade. NETRATE (Gomez-Rodriguez et al., 2011) and CONNIE (Myers & Leskovec, 2010) infer not only the network connectivity but either prior probabilities of infection or transmission rates of infection using convex optimization by considering all possible directed trees supported by each cascade.

The main innovation of this paper is to tackle the network inference problem as a submodular maximization problem in which we do not consider only the most probable directed tree as in NETINF but all directed trees supported by each cascade as in CONNIE and NETRATE. By considering all trees, we are able to infer a network more accurately than NETINF when the number of observed cascades is small compared to the network size and by using the greedy algorithm for submodular maximization in contrast to convex optimization, we are several order of magnitude faster than CONNIE and NETRATE. Therefore, we present a network inference algorithm that may be capable of inferring real networks in the order of hundred of thousands of nodes with a small number of observed cascades. This comes with a drawback, our algorithm does not infer prior probabilities of infection nor transmission rates but only the network connectivity. However, in practice, our algorithm provides a measure of *importance* for each edge of the network through the marginal gain that each edge provides.

Inferring how diffusion propagates over rapidly changing networks is crucial for a better understanding of the dynamics that govern processes taking place over information and social networks. In this context, scalability is a key point given the increasingly larger size of such networks and cascade data.

The remainder of the paper is organized as follows: in Section 2, we describe the model of diffusion and state the network inference problem. Section 3 shows an efficient approximation algorithm with *provable* near-optimal performance. Section 4 evaluates our method using synthetic and real data and we conclude with a discussion of our results in Section 5.

(a) Cascade c on G (b) Spanning trees induced by cascade c on G
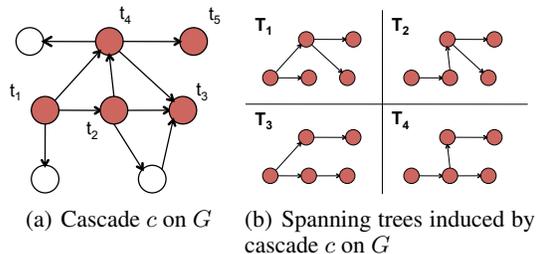
Figure 1. Panel (a) shows a cascade $\mathbf{t} = \{t_1, \ldots, t_5\}$ on network G, where $t_{i-1} < t_i$. Panel (b) shows all connected spanning trees induced by cascade $\mathbf{t}$ on G, *i.e.*, all possible ways in which a diffusion process spreading over G can create the cascade.

2. Problem formulation

In this section, we first describe the diffusion data our algorithm is designed for and continue revisiting the generative model of diffusion introduced recently by Gomez-Rodriguez et al. (2010). We conclude with a statement of the network inference problem.

Data. We observe a set C of cascades $\{\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}\}$ on a fixed population of N nodes. A cascade $\mathbf{t}^c := (t_1^c, \ldots, t_N^c)$ is simply a N-dimensional vector recording when nodes in the population get infected. We only observe the time t_i^c when a node i got infected but not who infected the node neither why it got infected. In each cascade, there are typically nodes that are never infected, with infection times that are arbitrarily long. We assume there is an underlying unobserved network G that nodes in the population belong to, and cascades propagate over it. Our aim is to discover this unknown network over which cascades originally propagated by using only the recorded infection times.

Pairwise transmission likelihood. We assume node j can infect node i with prior probability of transmission β. Now, consider that node j gets infected at time t_j and succeeds at infects node i at time t_i. We then assume that the infection time t_i depends on t_j through a pairwise transmission likelihood $f(t_i|t_j; \alpha_{j,i})$. As in previous studies of information propagation (Gomez-Rodriguez et al., 2010; 2011) and epidemiology (Wallinga & Teunis, 2004), we consider two well-known monotonic parametric models: exponential, $f(t_i|t_j; \alpha_{j,i}) \propto e^{-\alpha_{j,i} \cdot (t_i - t_j)}$, and power-law, $f(t_i|t_j; \alpha_{j,i}) \propto (t_i - t_j)^{-1-\alpha_{j,i}}$, and one non-monotonic parametric model: Rayleigh, $f(t_i|t_j; \alpha_{j,i}) \propto (t_i - t_j)e^{-\alpha_{j,i} \cdot (t_i - t_j)^2}$. Although we perform experiments in networks in which the transmission rate $\alpha_{j,i}$ of each edge can be different, in the remainder of the paper, for simplicity, we assume all transmission rates to be equal, $\alpha_{j,i} = \alpha$. Importantly, our algorithm does not depend on the particular choice of pairwise transmission likelihood and choosing more complicated parametric or non-parametric likeli-

hoods does not increase its computational complexity.

Likelihood of a cascade for a given tree. We assume that diffusion processes propagate as directed trees, *i.e.*, a node gets infected by action of a single node or parent. Then, for a given tree T and cascade $\mathbf{t}^c$, we can compute the likelihood of the cascade given the tree as follows:

$$f(\mathbf{t}^c|T) = \prod_{(u,v)\in E_T} f(t_v|t_u;\alpha), \qquad (1)$$

where E_T is the edge set of tree T. Considering a specific tree T for a cascade $\mathbf{t}^c$ means to set which edges have spread successfully the information. Therefore, given the tree T, we can compute the likelihood of the infection times of the nodes in the cascade $\mathbf{t}^c$ by using simply the pairwise transmission likelihood of each edge of the tree.

Probability of a tree in a given network. In order to compute the likelihood of a cascade $\mathbf{t}^c$ for a given tree T, we have considered the tree T to be given. We now compute the probability of a tree T in a network G as follows:

$$\begin{aligned} P(T|G) &= \prod_{(u,v)\in E_T} \beta \prod_{u\in V_T,(u,x)\in E\setminus E_T} (1-\beta) \\ &= \beta^q (1-\beta)^r, \end{aligned}$$

where V_T is the vertex set of tree T, E_T is the edge set of tree T, E is the edge set of the network G and $q = |E_T| = |V_T| - 1$ is the number of edges in T and counts the edges over which the diffusion process successfully propagated. For a particular cascade $\mathbf{t}^c$ and tree T, V_T is the set of nodes that belong to $\mathbf{t}^c$, *i.e.*, nodes where the infection time $t_i < \infty$. The first product accounts for the *active* edges in G, *i.e.*, edges that define the tree T, and the second product accounts for the *inactive* edges in G, *i.e.*, edges where the diffusion process did not spread. For simplicity, we assume the same prior probability of transmission β for every edge of the network G.

Likelihood of a cascade in a given network. Now, for a cascade $\mathbf{t}^c$, we consider all possible propagation trees T that are supported by the network G, *i.e.*, all possible ways in which a diffusion process spreading over G can create cascade $\mathbf{t}^c$:

$$f(\mathbf{t}^c|G) = \sum_{T\in\mathcal{T}_c(G)} f(\mathbf{t}^c|T)P(T|G), \qquad (2)$$

where $\mathbf{t}^c$ is a cascade and $\mathcal{T}_c(G)$ is the set of all the directed connected spanning trees on the subnetwork of G induced by the nodes that got infected in cascade $\mathbf{t}^c$, *i.e.*, $t_i \in \mathbf{t}^c : t_i < \infty$. Figure 1 illustrates the notion of a cascade and all the connected spanning trees T induced by its nodes.

All trees $T \in \mathcal{T}_c(G)$ employ the same vertex set V_T and $P(T|G)$ depends only the size of the vertex set V_T. Therefore, assuming the same prior probability of transmission β for every edge of the network, $P(T|G)$ is equal for all trees T on the subnetwork of G induced by the nodes that got infected in cascade $\mathbf{t}^c$ and we simplify Eq. (2):

$$f(\mathbf{t}^c|G) \propto \sum_{T\in\mathcal{T}_c(G)} \prod_{(u,v)\in E_T} f(t_v|t_u;\alpha). \qquad (3)$$

Now, assuming conditional independence between cascades given the network G, we compute the joint likelihood of a set C of cascades occurring in the network G as follows:

$$f(\mathbf{t}^1,\ldots,\mathbf{t}^{|C|}|G) = \prod_{\mathbf{t}^c\in C} f(\mathbf{t}^c|G). \qquad (4)$$

Network inference problem. Given a set of cascades $\{\mathbf{t}^1,\ldots,\mathbf{t}^N\}$, a prior probability of transmission β and a pairwise transmission likelihood $f(t_v|t_u;\alpha)$, we aim to find the network $\hat{G}$ such that

$$\hat{G} = \underset{|G|\leq k}{\operatorname{argmax}} f(\mathbf{t}^1,\ldots,\mathbf{t}^N|G), \qquad (5)$$

where the maximization is over all directed networks G of at most k edges.

3. Proposed algorithm

To the best of our knowledge, the optimization problem defined by Eq. (5) has been considered intractable in the past and proposed as an interesting open problem (Gomez-Rodriguez et al., 2010). We now show how to efficiently find a solution with *provable* sub-optimality guarantees by exploiting a natural diminishing returns property of the network inference problem: submodularity.

To evaluate Eq. (4), we need to compute Eq. (3) for each cascade $\mathbf{t}^c$, i.e., compute a sum of likelihoods over all possible connected spanning trees T induced by the nodes infected in each cascade. Although the number of trees can be super-exponential in the number of nodes in the cascade $\mathbf{t}^c$, this super-exponential sum can be performed in time polynomial in the number n of nodes in $\mathbf{t}^c$, by applying Kirchhoff's matrix tree theorem:

Theorem 1 (Tutte (1948)). *Given a directed graph W with non negative edge weights $w_{i,j}$, construct a matrix A such that $a_{i,j} = \sum_k w_{k,j}$ if $i = j$ and $a_{i,j} = -w_{i,j}$ if $i \neq j$ and denote the matrix created by removing any row x and column y from A as $A_{x,y}$. Then,*

$$(-1)^{x+y}\det(A_{x,y}) = \sum_{T\in\mathcal{T}(W)} \prod_{(i,j)\in T} w_{i,j}, \qquad (6)$$

where T is each directed spanning tree in W that starts in y.

Algorithm 1 Our network inference algorithm

Require: C, k
  $G \leftarrow \bar{K}$;
  while $|G| < k$ **do**
    for all $(j, i) \notin G : \exists \mathbf{t}^c \in C$ with $t_j < t_i$ **do**
      $\delta_{j,i} = 0, M_{j,i} \leftarrow \emptyset$;
      for all $\mathbf{t}^c : t_j < t_i$ **do**
        $w_c(m, n) \leftarrow$ weight of (m, n) in $G \cup \{(j, i)\}$;
        for all $t_m : t_m < t_i, m \neq j$ **do**
          $\delta_{c,j,i} = \delta_{c,j,i} + w_c(m, i)$;
        end for
        $\delta_{j,i} = \log(\delta_{c,j,i} + w_c(j, i)) - \log(\delta_{c,j,i} + 1)$
      end for
    end for
    $(j^*, i^*) \leftarrow \arg\max_{(j,i) \notin G} \delta_{j,i}$;
    $G \leftarrow G \cup \{(j^*, i^*)\}$;
  end while
  return G;

In our case, we compute Eq. (3) by setting $w_{i,j}$ to $f(t_j|t_i; \alpha)$ and computing the determinant in Eq. (6). We then compute Eq. (4) by multiplying the determinants of $|C|$ matrices, one for each cascade. For a fixed cascade $\mathbf{t}^c$, edges with positive weights form a directed acyclic graph (DAG) (only edges (i, j) such that $t_i < t_j$ have positive weights) and a DAG with a time-ordered labeling of its nodes has an upper triangular connectivity matrix. Thus, the matrix $A_{x,y}$ of Theorem 1, by construction, is also upper triangular. Fortunately, the determinant of an upper triangular matrix is simply the product of the elements of its diagonal and then,

$$f(\mathbf{t}^c|G) \propto \prod_{t_j \in \mathbf{t}^c} \sum_{t_i \in \mathbf{t}^c : t_i \leq t_j} f(t_j|t_i; \alpha).$$

This means that instead of using super-exponential time, we are now able to evaluate Eq. 4 in time $O(|C| \cdot N^2)$, where N is the size of the largest cascade, *i.e.*, the time required to build $A_{x,y}$ and compute the determinant for each of the $|C|$ cascades.

Until now, we have ignored the role of missed infections (Sadikov et al., 2011) or external sources as mass media (Katz & Lazarsfeld, 1955; Watts & Dodds, 2007) that can produce disconnected cascades. To overcome this point, we consider an additional node m that represents an external source that can infect *any* node u in a cascade. Therefore, we connect the external influence source m to every other node u with an ε-edge. Every node u can get infected by the external source m with an arbitrarily small probability ε. It is important to remark that adding the external source results in a tradeoff between false positives and false negatives when detecting cascades. The higher the value of ε, the larger the number of nodes that are as-

sumed to be infected by an external source.

Putting it all together, we include the additional node m in every cascade $\mathbf{t}^c$ and we set the likelihood of a diffusion process to spread from m to any node j in the cascade $\mathbf{t}^c$ to ε. We assume that $\varepsilon \leq f(t_j|t_i; \alpha)$ for any (i, j). We then define the improvement of log-likelihood for cascade $\mathbf{t}^c$ under graph G over an empty graph $\bar{K}$:

$$F(\mathbf{t}^c|G) = \sum_{t_j \in \mathbf{t}^c} \log \left(\sum_{t_i \in \mathbf{t}^c : t_i \leq t_j} w_c(i, j) \right), \quad (7)$$

where $w_c(i, j) = \varepsilon^{-1} f(t_j|t_i; \alpha) \geq 0$ for all natural likelihoods, $\sum_{i \in G : t_j \geq t_i} w_c(i, j) \geq 1$ and we assume that the ε-edges between m and all nodes in the cascade $\mathbf{t}^c$ exist also for the empty graph $\bar{K}$.

Finally, maximizing Eq. (5) is equivalent to maximizing the following objective function:

$$F_C(\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}|G) = \sum_{\mathbf{t}^c \in C} F(\mathbf{t}^c|G), \quad (8)$$

where G is the variable.

Efficient optimization. By construction, the empty graph $\bar{K}$ has score 0, $F_C(\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}|\bar{K}) = 0$, and the objective function F_C is non-negative monotonic, $F_C(\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}|G) \leq F_C(\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}|G')$, for any $G \subseteq G'$. Therefore, adding more edges to G never decreases the solution quality, and thus the complete graph maximizes F_C. However, in real-world scenarios, we are interested in inferring sparse graphs with a small number of edges. Thus, we would like to solve:

$$G^* = \underset{|G| \leq k}{\arg\max} \, F_C(\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}|G), \quad (9)$$

where the maximization is over all directed networks G of at most k edges. Naively searching over all k edge graphs would take time exponential in k, which is intractable. Moreover, finding the optimal solution to Eq. 9 is NP-hard:

Theorem 2. *The diffusion network inference problem defined by Eq. 9 is NP-hard.*

Proof. By reduction from the MAX-k-COVER problem (Khuller et al., 1999). □

While finding the optimal solution is hard, we will now show that F_C satisfies *submodularity* on the set of directed edges in G, a natural diminishing returns property, which will allow us to efficiently find a *provable* near-optimal solution to the optimization problem.

A set function $F : 2^W \to \mathbb{R}$ mapping subsets of a finite set W to the real numbers is *submodular* if whenever $A \subseteq$

(a) PR (Random, EXP)   (b) PR (Hierarchical, POW)   (c) PR (Core-periphery, RAY)

(d) Accuracy (Random, EXP)   (e) Accuracy (Hierarchical, POW)   (f) Accuracy (Core-periphery, RAY)
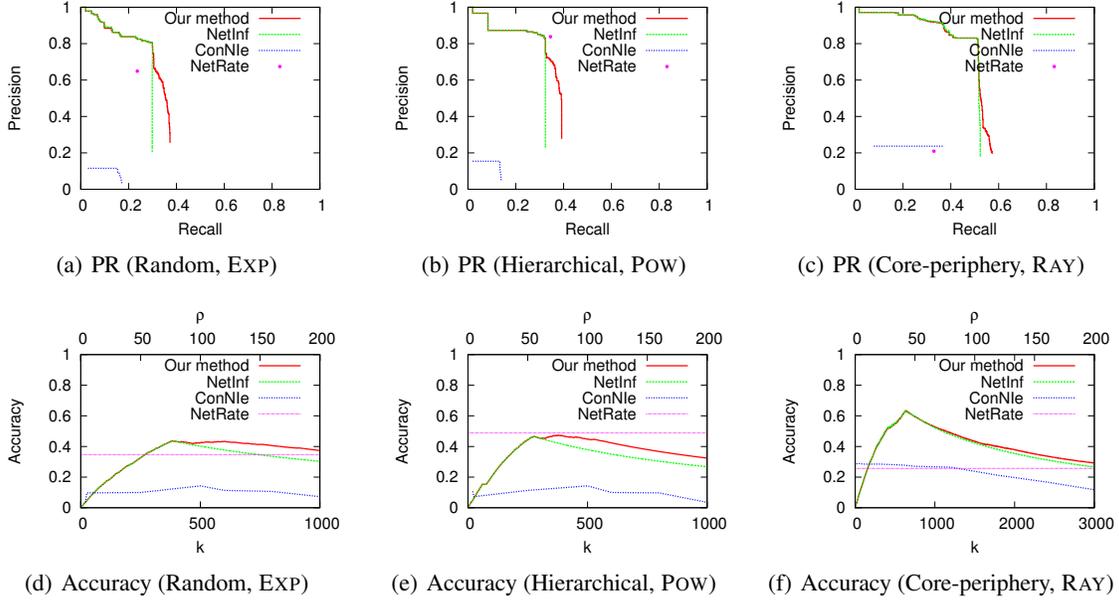
Figure 2. Panels (a-c) plot precision against recall (PR); panels (d-f) plot accuracy. To control the solution sparsity or precision-recall tradeoff, we sweep over k (number of edges) in our method and NETINF and over ρ (penalty factor) in CONNIE. NETRATE has no tunable parameters and therefore outputs a unique solution. (a,d): 1,024 node random Kronecker network with Rayleigh (RAY) model. (b,e): 1,024 node hierarchical Kronecker network with power-law (POW) model. (c,f): 1,024 node core-periphery Kronecker network with exponential (EXP) model. In all three networks, we recorded 200 cascades.

$B \subseteq W$ and $s \in W \setminus B$, it holds that $F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)$, i.e., adding s to the set A increases the score more than adding s to the set B. We have the following result:

Theorem 3. *Let V be a set of nodes, and C be a collection of cascades hitting the nodes V. Then $F_C(\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}|G)$ is a submodular function $F_C : 2^W \to \mathbb{R}$ defined over subsets $W \subseteq V \times V$ of directed edges.*

Proof. Fix a cascade $\mathbf{t}^c$, graphs $G \subseteq G'$ and an edge $e = (r, s)$ not contained in G'. We will show that $F(\mathbf{t}^c|G \cup \{e\}) - F(\mathbf{t}^c|G) \geq F(\mathbf{t}^c|G' \cup \{e\}) - F(\mathbf{t}^c|G')$. Let $w_{i,j}$ be the weight of edge (i,j) in G, and $w'_{i,j}$ in G'. Since $G \subseteq G'$, it holds that $w'_{i,j} \geq w_{i,j} \geq 0$. If (i,j) is contained in G and G', then $w_{i,j} = w'_{i,j}$. Let $T_{A,e} = \sum_{i \in A \setminus \{r\} : t_j \geq t_i} w_c(i, s)$. It holds that $T_{G',e} \geq T_{G,e}$. Hence,

$$F(\mathbf{t}^c|G \cup \{e\}) - F(\mathbf{t}^c|G) = \log\left(\frac{T_{G,e} + w_c(r,s)}{T_{G,e}}\right)$$
$$\geq \log\left(\frac{T_{G',e} + w_c(r,s)}{T_{G',e}}\right)$$
$$= F(\mathbf{t}^c|G' \cup \{e\}) - F(\mathbf{t}^c|G'),$$

proving submodularity of $F(\mathbf{t}^c|G)$. Now, since nonnegative linear combinations of submodular functions are sub-

modular, the function

$$F_C(\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}|G) = \sum_{c \in C} F(\mathbf{t}^c|G)$$

is submodular as well.   □

We now optimize $F_C(G)$ by using the *greedy algorithm*, a well-known efficient heuristic with provable performance guarantees. The algorithm starts with an empty graph $\bar{K}$ and it adds edges that maximize the *marginal gain* sequentially. That means, at iteration i we choose the edge $e_i = \text{argmax}_{e \in G \setminus G_{i-1}} F_C(G_{i-1} \cup \{e\}) - F_C(G_{i-1})$.

The algorithm stops once it has selected k edges, and returns the solution $\hat{G} = \{e_1, \ldots, e_k\}$. The greedy algorithm is guaranteed to find a set $\hat{G}$ which achieves at least a constant fraction $(1 - 1/e)$ (of the optimal value achievable using k edges (Nemhauser et al., 1978). Starting from the near-optimal solution given by the greedy algorithm, we could possibly improve the solution by applying a local search procedure.

As in the original NETINF formulation, our algorithm also allows for two speeds-up: localized updates and lazy evaluation (Algorithm 1). We can also obtain an on-line bound based simply on the submodularity of the objective function (Leskovec et al., 2007).

(a) Random   (b) Hierarchical   (c) Core-periphery
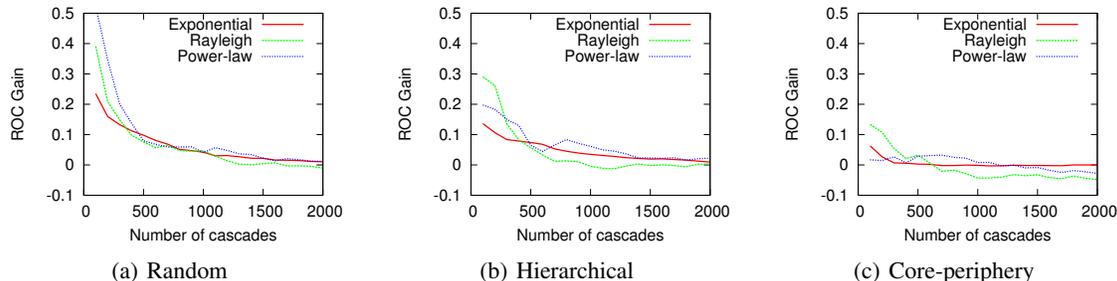
Figure 3. Gain in Area Under the ROC curve (AUC) of our method compared to NETINF vs number of cascades for (a) a random Kronecker network, (b) a hierarchical Kronecker network and (c) a core-periphery Kronecker network with 1,024 nodes and 1,024 edges for all three transmission models. Our method is able to more accurately infer a network for small number of cascades and it exhibits similar performance to NETINF for larger number of cascades.

4. Experimental evaluation

We evaluate our network inference algorithm in both synthetic and real networks. We use synthetic networks that aim to mimic the structure of social networks, and real information networks that are based on the MemeTracker dataset[1]. We compare our method in terms of precision, recall, accuracy and scalability with several state-of-the-art algorithms: NETINF, CONNIE and NETRATE. For the comparisons, we use the public domain implementations of these algorithms.

4.1. Experiments on synthetic data

Experimental setup. We first generate synthetic networks using two different well-known models of social networks: the Forest Fire (scale free) model (Barabási & Albert, 1999) and the Kronecker model (Leskovec et al., 2010), and set the pairwise transmission rates of the edges of the networks by drawing samples from $\alpha \sim U(0.5, 1.5)$. We then simulate and record a relatively small set of propagating cascades over each network using three different pairwise transmission likelihoods: exponential, power-law and Rayleigh. There are several reasons why we consider small set of cascades in comparison to the network size. First, all methods (including ours) assume that cascades propagate over a fixed network. Since social networks are highly dynamic (Backstrom & Leskovec, 2011), changing and growing rapidly, we can only expect to record a small number of cascades over a fixed network. Second, tracking and recording cascades is a difficult and expensive process (Leskovec et al., 2009). Therefore, it is desirable to develop network inference methods that work well with a small number of observed cascades.

Accuracy. We compare the inferred and true networks via three measures: precision, recall and accuracy. Precision

is the fraction of edges in the inferred network $\hat{G}$ present in the true network G^*. Recall is the fraction of edges of the true network G^* present in the inferred network $\hat{G}$. Accuracy is $1 - \frac{\sum_{i,j} |I(\alpha^*_{i,j}) - I(\hat{\alpha}_{i,j})|}{\sum_{i,j} I(\alpha^*_{i,j}) + \sum_{i,j} I(\hat{\alpha}_{i,j})}$, where $I(\alpha) = 1$ if $\alpha > 0$ and $I(\alpha) = 0$ otherwise. Inferred networks with no edges or only false edges have zero accuracy.

Figure 2 compares our method the precision, recall and accuracy of our method with for three different 1,024 node Kronecker networks: a random network (Erdős & Rényi, 1960) (parameter matrix $[0.5, 0.5; 0.5, 0.5]$), a hierarchical network (Clauset et al., 2008) ($[0.9, 0.1; 0.1, 0.9]$) and a core-periphery network (Leskovec et al., 2008) ($[0.9, 0.5; 0.5, 0.3]$), and 200 observed cascades. In terms of precision-recall, our method is able to reach higher recall values than NETINF, CONNIE and NETRATE, *i.e.*, it is able to discover more true edges from a small number of cascades than other methods. For recall values that are reachable using NETINF, our method and NETINF offer a very similar precision value. Our methods allows for higher recall in comparison with NETINF because it gets exhausted[2] later for considering all possible trees per cascade instead of only the most probable one. In terms of accuracy, our method outperforms NETINF for more than half of their outputted solutions, and matches the remaining ones. CONNIE and NETRATE's accuracy is typically significantly lower. However, NETRATE is able to beat all other methods for the hierarchical Kronecker network. If we compare with previous studies (Myers & Leskovec, 2010), the performance of CONNIE seem to degrade the most due to the limited availability in cascades and perhaps the variable transmission rates across the networks (as reported previously in Gomez-Rodriguez et al. (2011)).

Performance vs. cascade coverage. Intuitively, the more

---

[1]Data available at `http://memetracker.org`

[2]A greedy method (ours and NETINF) gets exhausted at iteration k when there are not any more edges with marginal gain larger than zero.
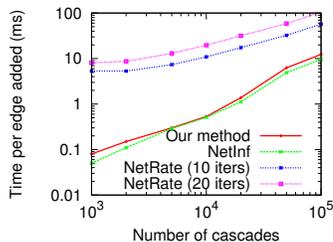
Figure 4. Average running time per edge added against number of cascades. We used a 1,024 node random Kronecker with exponential transmission model.

(a) Precision-recall   (b) Accuracy

Figure 5. Real data. Panel (a) plots precision-recall and panel (b) accuracy on a 1,000 node hyperlink network with 10,000 edges using 1,000 cascades and a power-law model. To control the solution sparsity or precision-recall tradeoff, we sweep over k (number of edges) in our method and NETINF and over ρ (penalty factor) in CONNIE. Our method beats others for the majority of their outputted solutions.

cascades we observe, the more accurately *any* algorithm infers a network. Actually, when the number of cascades is large in comparison to the network size, we expect differences in performance among methods become negligible. Figure 3 plots the gain in Area Under the ROC curve (AUC) for our method in comparison with NET-INF, $(\mathrm{AUC}_{\text{our method}} - \mathrm{AUC}_{\text{NETINF}})/\mathrm{AUC}_{\text{NETINF}}$, against number of observed cascades for several Kronecker networks and transmission models ($\beta = 0.5$ and $\alpha \sim U(0.5, 1.5)$ in all models). We observe that the difference in performance between our method and NETINF is greater for small number of cascades and for a large enough number of cascades, both methods perform similarly or NETINF slightly outperforms our method.

Scalability. Figure 4 plots the average computation time per edge added against number of cascades. Since NET-RATE is not greedy and instead solve a convex program for each node in the network, we divided their total running times by the number of edges that our method added until getting exhausted (until no edge has marginal gain greater than zero). We used the publicly available implementations of our algorithm and NETINF, both coded in C++. To carry out a fair comparison with NETRATE, we have developed a projected full gradient descend C++ implementation of NETRATE, which is considerably faster than the publicly available Matlab implementation (that uses the CVX convex solver), and we run 10 and 20 iterations of full gradient descend (remarkably, even running one single iteration was slower than NETINF and our method). We do not report running times for CONNIE since the publicly available code is a Matlab implementation (that uses the SNOPT solver) and probably slower than a C++ implementation. Our method and NETINF are approximately one order of magnitude faster than NETRATE. Finally, note that the running time of our algorithm does not depend on the network size but the number of cascades and cascade size. As an experimental validation, we run our algorithm in two networks with $100,000$ and $200,000$ nodes and an average of two edges per node using $10,000$ cascades and our algorithm took only 10.12 ms and 12.14 ms per edge added.

4.2. Experiments on real data

Experimental setup. We use the publicly available MemeTracker dataset, which contains more than 172 million news articles and blog posts from 1 million online sources (Leskovec et al., 2009). Sites publish pieces of information and use hyperlinks to refer to their sources, which are other sites that published the same or closely related pieces of information. Therefore, we use hyperlinks to trace information propagation over blogs and media sites. A hyperlink cascade is simply a collection of time-stamped hyperlinks between sites (in blog or news media posts) that refer to the same or closely related pieces of information. We record one hyperlink cascade per piece or closely related pieces of information. We extract the top 1,000 media sites and blogs with the largest number of documents, 10,000 hyperlinks and 500 longest hyperlink cascades. We create a ground truth network G which contains an edge between a site u and a site v if there is at least a site post in the site u that links to a post on the site v. We then infer a network $\hat{G}$ from the hyperlink cascades and evaluate precision, recall and accuracy with respect to G. We consider a power law pairwise transmission likelihood. Note that we trace the flow of information and create a ground truth network using hyperlinks because we are interested in a quantitative evaluation of our method in comparison with the state of the art. For richer qualitative insights, cascades based on short textual phrases should be considered, but that goes beyond the scope of this paper.

Accuracy. Figure 5 shows precision, recall and accuracy of our method in comparison with NETINF, CONNIE and NETRATE. Our method reaches higher recall values than any other methods. In terms of accuracy, it beats others for the majority of their outputted solutions. As in the synthetic experiments, the shortage of recorded cascades degrades CONNIE's performance dramatically.

5. Conclusions

We have developed an efficient approximation algorithm with provable near-optimal performance that solves an open problem on network inference from diffusion traces (or cascades) first introduced by Gomez-Rodriguez et al. (2010). In our work, for each observed cascade we consider all possible ways in which a diffusion process spreading over the network can create the cascade, in contrast with NETINF, that considers only the most probable way (tree).

Perhaps surprisingly, despite considering all trees, we show experimentally that the running time of our method and NETINF are similar, and they are several orders of magnitude faster than alternative network inference methods based on convex programming as NETRATE and CONNIE. Moreover, our algorithm typically outperforms NETINF, NETRATE and CONNIE in terms of precision, recall and accuracy in highly dynamic networks in which we only observe a relatively small set of cascades before they change significantly.

References

Backstrom, L. and Leskovec, J. Supervised random walks: predicting and recommending links in social networks. In *WSDM '11*, 2011.

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

Brockmann, D., Hufnagel, L., and Geisel, T. The scaling laws of human travel. *Nature*, 439(7075):462–465, 2006. ISSN 0028-0836.

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

Erdős, P. and Rényi, A. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Science*, 5:17–67, 1960.

Gomez-Rodriguez, M., Leskovec, J., and Krause, A. Inferring Networks of Diffusion and Influence. In *KDD '10: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pp. 1019–1028, 2010.

Gomez-Rodriguez, M., Balduzzi, D., and Schölkopf, B. Uncovering the Temporal Dynamics of Diffusion Networks. In *ICML '11*, 2011.

Katz, E. and Lazarsfeld, P.F. *Personal influence: The part played by people in the flow of mass communications*. Free Press, 1955.

Kempe, D., Kleinberg, J. M., and Tardos, É. Maximizing the spread of influence through a social network. In *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 137–146, 2003.

Khuller, S., Moss, A., and Naor, J. The budgeted maximum coverage problem. *Information Processing Letters*, 70 (1):39–45, 1999.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. Cost-effective outbreak detection in networks. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 420–429, 2007.

Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. Statistical properties of community structure in large social and information networks. In *WWW '08*, 2008.

Leskovec, J., Backstrom, L., and Kleinberg, J. Memetracking and the dynamics of the news cycle. In *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

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

Myers, S. and Leskovec, J. On the Convexity of Latent Social Network Inference. In *NIPS '10*, 2010.

Nemhauser, GL, Wolsey, LA, and Fisher, ML. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

Sadikov, S., Medina, M., Leskovec, J., and Garcia-Molina, H. Correcting for missing data in information cascades. In *WSDM '11: ACM International Conference on Web Search and Data Mining*, 2011.

Snowsill, T.M., Fyson, N., De Bie, T., and Cristianini, N. Refining causality: who copied from whom? In *KDD '11: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 466–474, 2011.

Wallinga, J. and Teunis, P. Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures. *American Journal of Epidemiology*, 160(6):509–516, 2004.

Watts, Duncan J. and Dodds, Peter S. Influentials, networks, and public opinion formation. *Journal of Consumer Research*, 34(4):441–458, 2007.