Learning from Contagion (Without Timestamps)

Kareem Amin AKAREEM@SEAS.UPENN.EDU
Hoda Heidari HODA@SEAS.UPENN.EDU
Michael Kearns MKEARNS@CIS.UPENN.EDU
Computer and Information Science, University of Pennsylvania

Abstract

We introduce and study new models for learning from contagion processes in a network. A learning algorithm is allowed to either choose or passively observe an initial set of seed infections. This seed set then induces a final set of infections resulting from the underlying stochastic contagion dynamics. Our models differ from prior work in that detailed vertex-by-vertex timestamps for the spread of the contagion are not observed. The goal of learning is to infer the unknown network structure. Our main theoretical results are efficient and provably correct algorithms for exactly learning trees. We provide empirical evidence that our algorithm performs well more generally on realistic sparse graphs.

1. Introduction

We present new techniques for *learning from contagion* in networks. Our motivation is settings in which we are able to select or observe an initial "seed" set of infected vertices, and only at some later time, observe the set of subsequent infections resulting from some underlying stochastic contagion process on the network. Much classical work on network diffusion processes focuses on characterizing how contagions spread across a known network. Our aim, on the other hand, is to develop a more comprehensive theory for inferring *unknown* network structure from *observed* contagion behavior.

As an example, consider epidemiological studies, where an initial set of infections are observed in a population, and after a while, new infections develop. One might hope to learn the underlying social interactions within the population so as to better prevent future contagion. In direct marketing, a marketer might advertise to a group of potential customers. After some time, certain members of the population adopt a product.[1] Inferring edges in the network can help optimize future campaigns.

Only recently has attention been given to the problem of inferring network structure from contagion/diffusion processes (see Section 1.1). These recent works have assumed that, rather than observing only the initial and final infection sets, the learner is given temporal information describing the exact order in which infections take place on the network. This is a reasonable assumption when considering, for example, contagion in online social networks, where timestamped content is readily available. However, for many other natural contagion processes, accurate transcripts of how the contagion evolved are often unavailable; instead one only observes snapshots of infection states.

For the underlying contagion process, we examine the well-studied *independent cascade model* (Kempe et al., 2003; Goldenberg et al., 2001a;b). We consider both the case where the learner has *active seed queries*, or the weaker model of only *passive seed observations*. Our main results provide efficient algorithms for learning tree-like structures for both active and passive seed selection, along with impossibility results showing that some of our sample-size dependencies are necessary. We then investigate the important and natural extension of these results to cyclical graphs. We show that a natural extension of the algorithms we propose can be utilized to learn the structure (exactly or with low error) so long as the underlying network is sufficiently sparse. When the network is too dense, the presence of many short cycles in a network presents challenges in our models that bear a high-level similarity to those of problems such as inference in Bayesian networks — the short cycles create many paths via which vertices can influence each other, conflating the sources for large-scale contagion and making learning difficult. However, we hope that some of the analytical and algorithmic tools we develop may be useful for even richer classes of networks.

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

[1]In this example the marketer actively selects the seed set, instead of passively observing it.

1.1. Related Work

Our work mainly concerns a well-studied model for contagion, the independent cascade model (Kempe et al., 2003; Goldenberg et al., 2001a;b) Only recently has there been work on learning network structure from contagion (Rodriguez & lkopf, 2012; Myers & Leskovec, 2010; Rodriguez et al., 2011; Abrahao et al., 2012; Gomez-Rodriguez et al., 2010; Du et al., 2012; Netrapalli & Sanghavi, 2012), primarily (with slight variations) in the independent cascade model.

As previously mentioned, a key assumption in these works is that the learner observes a sequence of cascade processes with timestamps, or the exact order in which vertices became infected. Gathering such information about the infection times is not always possible. For example, when monitoring the effectiveness of a viral marketing campaign, a firm might survey a population some time into the campaign in order to assess its effectiveness, but it is unlikely that this would reveal information on the precise order respondents adopted the product. In contrast, we assume that we have access to data (S_i, A_i), where S_i is some initial seed set of infections, and A_i are infections eventually resulting from the contagion process when the vertices in S_i were initially infected.

This weaker assumption vastly changes the types of techniques that must be employed in order to infer network structure. In particular, the learner can no longer directly attribute the infection of some vertex to some previously infected vertex by watching the contagion unfold. To the best of our knowledge (Gripon & Rabbat, 2013) is the only work that addresses the problem of learning the network structure without such transcripts. There the learner is allowed access to all triples $\{a, b, c\}$ where $\{a, b, c\}$ are path-connected vertices.

Furthermore, for the learning objective, unlike (Rodriguez & lkopf, 2012; Myers & Leskovec, 2010; Rodriguez et al., 2011) we do not seek to maximize the likelihood of the observed data, rather, similar to (Abrahao et al., 2012), our goal is to (exactly or approximately) reconstruct the underlying network.

As mentioned in the introduction, we study two ways of seeding, or initiating, the infection: the active model, in which the learner can choose the seeds, and the passive model in which the seed sets are randomly sampled from a distribution. Both the active (see (Abrahao et al., 2012; Myers & Leskovec, 2010)) and the passive model (see (Netrapalli & Sanghavi, 2012; Gomez-Rodriguez et al., 2010)) have been studied in prior work.

Finally, we highlight that our results are largely combinatorial in nature, whereas the previous literature uses convex programming (Myers & Leskovec, 2010; Rodriguez et al., 2011; Du et al., 2012; Netrapalli & Sanghavi, 2012), or submodularity (Rodriguez & lkopf, 2012; Gomez-Rodriguez et al., 2010) to approximate an optimal network.

There are a number of other papers that are remotely related to our work: (Vert & Yamanishi, 2005) examine the problem of network inference in the context of supervised learning, where part of the graph is revealed to the learner and the goal is to learn edges that connect "similar" vertices to each other. (Lippert et al., 2009) propose an unsupervised kernel-based method instead.

2. Models

The broad setting we consider is the following: There is an unknown network structure $G(V, E)$ in which an initial *seed set* $S \subseteq V$ becomes infected; the contagion subsequently spreads according to the rules of some underlying (possibly stochastic) contagion model with parameters θ, and generates the final infected set $A(S) \subseteq V$ observed at some later time. The network structure, contagion model, and seed infection set define a distribution $\mathcal{D} = \mathcal{D}(S, G, \theta)$ on 2^V, and a draw $A \sim \mathcal{D}$ specifies the subset of V that becomes infected after the contagion has run its course.

Our results examine the stochastic *independent cascade* (Kempe et al., 2003; Goldenberg et al., 2001a;b) model of contagion.[2] In the independent cascade model each edge (u, v) has an associated infection probability $p_{(u,v)}$, and the dynamics of the contagion are as follows: when a vertex u first becomes infected, it infects each of its neighbors, say $v \in N(u)$, with probability $p_{(u,v)}$. We say the edge (u, v) is *flipped* once this probability $p_{(u,v)}$ trial is conducted. Regardless of whether v is infected as a result, the edge (u, v) can never be flipped again. An alternate way of viewing the model is that we first flip each edge in G with the appropriate probability, and delete edges for which the trial fails, resulting in a subgraph of G. Then any vertex in the same connected component of this subgraph as some seed vertex becomes infected.

In the *active* seed selection model, a learner is permitted to choose an adaptive sequence of seed sets $S_1, ..., S_N$ each of which generates a resulting set of infections $A_i = A(S_i) \sim \mathcal{D}(S_i, G, \theta)$. In the *passive* seed selection model, the learner is not permitted to choose the seed sets, but rather they are chosen randomly from a distribution $\mathcal{P}$. In the passive model, we will need to refer to the distribution which is induced by first drawing $S \sim \mathcal{P}$ then $A \sim \mathcal{D}(S, G, \theta)$. For this we use the shorthand $\mathcal{D}(\mathcal{P}, G, \theta)$. Broadly speaking, the active seed model is more appropriate for settings such as viral marketing, where the learner

[2]While similar results can be obtained for the linear threshold model of contagion, due to space considerations we only focus on the independent cascade model.

can repeatedly target different populations for infections via promotions, advertising, or give-aways, while the more challenging passive seed model is better suited for settings in which Nature determines the initial infections, as in the spread of actual disease such as annual flu cycles. Note that in either case we assume the underlying contagion model is known, but the actual parameters of the model (e.g. the probabilities $p_{(u,v)}$) are not.

The main problem we consider is that of exactly detecting the underlying network structure.[3] Doing so efficiently will in general require that the true or target graph belong to some restricted class $\mathcal{G}$. By "efficient" we mean the standard polynomial dependence on $|V|$, and on other parameters that we shall discuss at the appropriate place. Within this framework, we consider classes of graphs that are restricted to be trees or "near" trees in the sense of having few or limited cycles. We also demonstrate empirically that similar algorithms provide us with low error rates when learning sparse network structures.

3. Exactly Learning Trees with Active Seeds

Our first result gives an efficient algorithm for exactly learning the structure of an arbitrary undirected tree in the active seed model. The sample size required[4] (and running time) will depend inversely on $1/\Delta$, where we define $\Delta = \min_{u,v} \min\{1 - p_{(u,v)}, p_{(u,v)}\}$. This dependence is clearly necessary for any algorithm — if the $p_{(u,v)}$ can be arbitrarily small, infections will never be observed, and exact learning of the structure is impossible.

Theorem 1. *In the active seed selection model and independent cascade contagion, any tree structure is exactly learnable from samples $\{(S_i, A_i(S_i))\}_{i=1}^M$, where M is polynomial in $\frac{1}{\Delta}$, $|V|$ and $\log(\frac{1}{\delta})$.*

We present an algorithm that will only select seed sets of size one (singleton seeds). The algorithm generates $M = m|V|$ observations by selecting each vertex u as the seed m times (m will be determined by the analysis).[5]

We then define $R_u(v) = \{i \mid v \in A_i \wedge S_i = \{u\}\}$. Thus $R_u(v)$ denotes the rounds in which v was infected given that u was the seed vertex. We shall show with high probability, the containment relationships between the $R_u(v)$ uniquely determines the tree structure. The proof of Theorem 1 follows as a consequence.

[3]Our experimental results investigate the relaxation to approximating the structure. It would also be natural to consider approximating the distribution $\mathcal{D}(S_i, G, \theta)$, for which we have results outside the scope of this paper.

[4]Note that throughout this work, we have not attempted to optimize the required sample sizes.

[5]Thus the approach will also suffice in the passive setting when $\mathcal{P}$ is supported on singleton sets of V.

Let $N(u)$ denote the set of vertices adjacent to vertex u.

Lemma 1. *Suppose $v \notin N(u)$ and $m \geq \frac{1}{\Delta^2}\log(1/\delta)$. Then with probability at least $1-\delta$ there exists a v' where $R_u(v) \subsetneq R_u(v')$.*

Proof. If v is not a neighbor of u, then there exists some neighbor of u, v', along the path from u to v. Given that the underlying network is a tree, every time v is infected, v' must also be infected, establishing that $R_u(v) \subseteq R_u(v')$. For the strict inclusion, let v'' be the immediate neighbor of v' that isn't u (but is possibly v). Given that the underlying network is a tree, every time u is selected as the seed vertex, the probability that the contagion reaches v' but fails to reach v'' is $p_{uv'}(1 - p_{v'v''}) \geq \Delta^2$. Thus, with probability at least Δ^2, v' is infected but not v. The probability that the infection reaches both v' and v for all m rounds in which u is the seed vertex is therefore at most $(1-\Delta^2)^m \leq \exp(-\Delta^2 m) \leq \delta$ for $m = \frac{1}{\Delta^2}\log(1/\delta)$, establishing the lemma. □

Lemma 2. *Suppose $v, v' \in N(u)$ and $m \geq \frac{1}{\Delta^2}\log(2/\delta)$. Then neither $R_u(v) \subseteq R_u(v')$ nor $R_u(v') \subseteq R_u(v)$ with probability at least $(1-\delta)$.*

Proof. Given that the underlying network is a tree, if u is the seed vertex, the probability that the contagion reaches v but not v' is $p_{(u,v)}\left(1 - p_{(u,v')}\right) \geq \Delta^2$. Standard arguments yield that $m \geq \frac{1}{\Delta^2}\log(1/\delta)$ suffices for this to occur on at least one round, thereby ensuring that $R_u(v) \not\subseteq R_u(v')$. The symmetric argument, swapping v and v', and a union bound yields the lemma. □

We say that v is *maximally infected* by u if there does not exist a v' such that $R_u(v) \subseteq R_u(v')$. If $m = O\left(\frac{1}{\Delta^2}\log(|V|/\delta)\right)$, then with probability at least $(1-\delta)$, for all u, v pairs, v is maximally infected by u if and only if $v \in N(u)$. This suggests a clear procedure for detecting the neighborhood of each vertex u. Repeating this for all vertices, we can exactly learn the tree with high probability.

4. Exactly Learning Trees with Passive Seeds

While the algorithm for learning in the active model required non-trivial book keeping, the learner in that setting is greatly aided by the fact that all infections observed in a given round must have originated from a particular seed. In the passive model, this is no longer the case – the affects of seeding a particular vertex may never be seen in isolation. Thus attributing any infection in A_i to any particular vertex in S_i is more difficult.

As for the seed distribution $\mathcal{P}$, we obviously cannot hope to exactly learn for arbitrary distributions, since this would include pathological cases such as when $S_i = \{V\}$ with probability one. Therefore in what follows we assume that $\mathcal{P}$ belongs to the family of product distributions where each $u \in S_i$ becomes initially infected independently with probability q_u for unknown $0 < q_u < 1$[6]. We will once again let Δ be the smallest value among the $p_{u,v}$, q_u,$1-p_{u,v}$ and $1-q_u$.

4.1. Characterizing Lifts

Our algorithm will work by observing the relation that a particular vertex's infection has on likelihood of infection at other locations in the tree. To this end, we will define the *lift* u has on v as:

$$L(v \mid u) = P(v \in A_i \mid u \in S_i) - P(v \in A_i) \qquad (1)$$

In other words, this lift is the increase in the probability that v is infected from conditioning on u's presence in the (passive) seed set. Before describing the algorithm, we will record some useful facts about lifts.

First notice that the outcome of the independent cascade model is determined by the outcomes of $|V|+|E|$ Bernoulli random variables. $|V|$ coins are flipped to determine which vertices belong to the seed set S_i, after which the $|E|$ edges of G are flipped to determine the outcome A_i of the contagion process. We say that an edge (u, v) is "active" if the random variable corresponding to it is equal to 1, and thus contagion is allowed to pass freely through that edge. Thus, if (u, v) is active, either both u and v are infected, or neither is infected. We define the active component containing u to be the subgraph of T which contains u and any vertex v for which all edges in the path $u-v$ are active. Once again, observe that all vertices in the active component containing u are either infected or uninfected.

Our first lemma gives a decomposition for $L(v \mid u) = \phi(u, v)\psi(u, v)$ in terms of two quantities ϕ, ψ. The first of these quantities is $\phi(u, v) = \prod_{(w,w')\in u-v} p_{(w,w')}$, where the product is over the edge set of the path $u - v$. In other words, $\phi(u, v)$ is simply the probability that the $u - v$ path is active. We say that $s - w$ is an **infecting tributary for** $u - v$ if $s \in S_i$, $w \in u - v$ and the path $s - w$ is active (we consider the trivial path $s - s$ when both $s \in u - v$ and $s \in S_i$ to be an infecting tributary for $u - v$ as well). Let $\mathcal{E}(u, v)$ be the event that there is no infecting tributary for $u - v$ and $\psi(u, v) = P(\mathcal{E}(u, v))$.

Lemma 3. *For any* u, v, $L(v \mid u) = \phi(u, v)\psi(u, v)$.

Proof. Fix u and v. Let $\mathcal{C}(u, v)$ (for "Connected") be the event that the $u - v$ path is active. Let $\mathcal{D}(u, v)$ (for "Disjoint") be the event the $u-v$ path is not active (equivalently, u and v belong to disjoint active components).

[6] We leave as an open question whether we can learn under more general assumptions on the family of seed set distributions.

We first derive an expression for $P(v \in A \mid u \in S)$:

$$\begin{aligned} P(v \in A \mid u \in S) &= P(\mathcal{D}(u,v) \mid u \in S)P(v \in A \mid u \in S, \mathcal{D}(u,v)) \\ &+ P(\mathcal{C}(u,v) \mid u \in S)P(v \in A \mid u \in S, \mathcal{C}(u,v)) \\ &= P(\mathcal{D}(u,v))P(v \in A \mid u \in S, \mathcal{D}(u,v)) \\ &+ P(\mathcal{C}(u,v))P(v \in A \mid u \in S, \mathcal{C}(u,v)) \\ &= P(\mathcal{D}(u,v))P(v \in A \mid u \in S, \mathcal{D}(u,v)) \\ &+ P(\mathcal{C}(u,v)) \\ &= P(\mathcal{D}(u,v))P(v \in A \mid \mathcal{D}(u,v)) + P(\mathcal{C}(u,v)) \end{aligned}$$

This first two equalities above follow from the law of total probability, and the fact that the event $u \in S$ is independent of $\mathcal{C}(u, v)$. The next equality follows by observing that $v \in A$ with certainty if $u \in S$ and the $u - v$ path is active. The final line follows from the conditional independence of the events $u \in S$ and $v \in A$ given $\mathcal{D}(u, v)$. Writing

$$\begin{aligned} P(v \in A) &= P(\mathcal{D}(u,v))P(v \in A \mid \mathcal{D}(u,v)) \\ &+ P(\mathcal{C}(u,v))P(v \in A \mid \mathcal{C}(u,v)) \end{aligned}$$

and applying the above, we conclude that:

$$\begin{aligned} L(v \mid u) &= P(v \in A \mid u \in S) - P(v \in A) \\ &= P(\mathcal{C}(u,v)) - P(\mathcal{C}(u,v))P(v \in A \mid \mathcal{C}(u,v)) \\ &= P(\mathcal{C}(u,v))(1 - P(v \in A \mid \mathcal{C}(u,v))) \\ &= \phi(u,v)P(v \notin A \mid \mathcal{C}(u,v)) \end{aligned}$$

Conditioned on $u - v$ being active, the probability that $v \notin A$ is precisely the probability that there are no infecting tributaries for $u - v$, completing the proof. □

The previous lemma gives several insights into $L(\cdot \mid \cdot)$. For example, we see that the function is symmetric in its arguments. Also note that lifts are monotonic along paths: for any $u-v$ path and w along that path, u provides a better lift to w than to v. A useful special case is recorded in the next lemma.

Lemma 4. *Let* $u - v$ *be a path of length greater than* 1*, and let* (w, w') *be any edge along that path.* $L(v \mid u) \leq (1 - \Delta)L(w \mid w')$.

Proof. Appealing to Lemma 3, we see that $L(v \mid u) = \phi(u, v)\psi(u, v)$. Recall that for every edge e, $p_e \leq 1 - \Delta$. Since $u - v$ contains at least one edge not equal to (w, w'), by definition of ϕ, $L(v \mid u) \leq (1 - \Delta)\phi(w, w')\psi(u, v)$.

Recalling the definition of ψ, we see that $\mathcal{E}(u, v)$ implies $\mathcal{E}(w, w')$. If there are no infecting tributaries for $u - v$ there cannot be any infecting tributaries for $w - w'$. Thus $\psi(u, v) \leq \psi(w, w')$, and $L(v \mid u) \leq (1 - \Delta)\phi(w, w')\psi(w, w')$. Applying Lemma 3 once more completes the proof. □

4.2. Algorithm

Armed with our understanding of lifts, we can now give an algorithm for learning an unknown tree T under passive infections. The central observation of the previous section is that, when w is a vertex along a $u-v$ path, then u must provide a better lift to w than to v. We might have hoped that a vertex u gives all its highest lifts to its neighbors. Unfortunately, this is not true. u could, for example, be at the head of a path $u-v$ for which every edge $e \in u-v$ has p_e very close to 1. Thus the presence of an infection at u has a large effect on the infection rate of all the vertices along $u-v$. In contrast, some of u's immediate neighbors w might already be prone to infection, or may have small $p_{(u,v)}$, and therefore conditioning on u being seeded does not change the observed infection-rate of w by much. Ultimately, we will have to exploit the monotonicity of $L(\cdot \mid \cdot)$ along paths, established in the previous section.

In what follows, we assume that the algorithm has observed M iid samples (S_i, A_i), from which it has derived estimates $\hat{L}(v \mid u)$ for each pair of vertices u, v. The exact nature of this sampling will be covered subsequently. For the sake of intuition, suppose that these estimates are perfect. The algorithm works by growing a forest of connected components of T. Given such a component C, a $u \in C$ and a $v \notin C$, if u and v are not neighbors then there is an edge $(x, y) \neq (u, v)$ where the $u-v$ path crosses out of the component C (i.e. $x \in C$, but $y \notin C$). Lemma 4 tells us that $L(v \mid u) \leq L(y \mid x)$. Thus, by taking $(u^*, v^*) = \arg\max_{u \in C, v \notin C} L(v \mid u)$, the algorithm is guaranteed to find an edge of the network. We present the pseudocode for the algorithm below.

Algorithm 1 Algorithm for exactly learning trees under passive seeds and independent cascade.

```
1: Input: Estimates $\hat{L}(\cdot \mid \cdot)$, vertex set $V$
2: % Begin with singleton components $C_i = \{u\}$ for each $u \in V$
3: % and the empty edge-set.
4: Set Components $= \{\{v\} \mid v \in V\}$.
5: Set $\hat{E} = \emptyset$
6: % Iterate until we have a single component.
7: while $|\mathcal{C}| \neq 1$ do
8:     for $C_i \in \mathcal{C}$ do
9:         % Discover an edge.
10:        $(u^*, v^*) = \arg\max_{u \in C_i, v \notin C_i} \hat{L}(v \mid u)$
11:        Add $(u^*, v^*)$ to $E$.
12:    end for
13:    Set Components equal to connected components of $(V, \hat{E})$
14: end while
15: Return $(V, \hat{E})$
```

4.3. Analysis

It is not difficult to show that this algorithm requires $O\left(|V|^2 \log |V|\right)$ computation time. The following theorem establishes the correctness of the algorithm given sufficiently good estimates. First, let us define notation for the smallest lift between two neighboring vertices: $L_{\min} = \min_{(u,v) \in E} L(v \mid u)$.

Theorem 2. *Suppose for every u, v, $|\hat{L}(v \mid u) - L(v \mid u)| < \frac{\Delta L_{\min}}{2}$. Then Algorithm 1 returns $\hat{E} = E$ in time $O\left(|V|^2 \log |V|\right)$.*

Proof. Fix an arbitrary subgraph C_i of the true network (V, E). To prove correctness, it suffices to show that for this choice of C_i line 10 returns (u^*, v^*) only if $(u^*, v^*) \in E$. Suppose for the sake of contradiction that this is not the case. Then the true $u^* - v^*$ path contains an edge (w, w') where $w \in C_i$ and $w' \notin C_i$. Lemma 4 tells us that $L(w \mid w') - L(v \mid u) \geq \Delta L(w \mid w') \geq \Delta L_{\min}$. By assumption, $|\hat{L}(v \mid u) - L(v \mid u)| < \frac{\Delta L_{\min}}{2}$, and therefore $\hat{L}(w \mid w') - \hat{L}(v \mid u) > 0$. Since $w \in C_i, w' \notin C_i$, (u^*, v^*) could not have been the argmax of line 10, establishing the contradiction. □

Finally, we provide an upper bound on the number of samples M needed in order to satisfy the hypothesis of Theorem 2 with high probability. Let $\hat{n}_A(v) = \sum_{i=1}^M \mathbf{1}(v \in A_i)$, $\hat{n}_{S,A}(u, v) = \sum_{i=1}^M \mathbf{1}(u \in S_i, v \in A_i)$, and $\hat{n}_S(u) = \sum_{i=1}^M \mathbf{1}(u \in S_i)$. Taking $\hat{L}(v \mid u) = \hat{n}_{S,A}(u, v)/\hat{n}_S(u) - \hat{n}_A(v)/M$ lets us state the following theorem.

Theorem 3. $M = O\left(\frac{1}{L_{\min}^2 \Delta^4} \log(|V|/\delta)\right)$ *suffices so that for any u, v, $|L(v \mid u) - \hat{L}(v \mid u)| < \Delta L_{\min}/2$ with probability at least $1-\delta$. Furthermore, as a consequence of this fact and Theorem 2, Algorithm 1 returns $\hat{E} = E$ in time $O\left(|V|^2 \log |V|\right)$ with probability at least $1-\delta$.*

Proof. Fix a u, v and $\epsilon > 0$. There exists a $C > 0$ such that taking $M = \frac{C}{\epsilon^2} \log(16|V|^2/\delta)$, guarantees with probability at least $1 - \frac{\delta}{2|V|^2}$: (1) $(1-\epsilon)P(u \in S) < \frac{1}{M}\hat{n}_S(u) < (1+\epsilon)P(u \in S)$, (2) $|\frac{1}{M}\hat{n}_{S,A}(u, v) - P(u \in A, v \in S)| \leq \epsilon$, and (3) $|\frac{1}{M}\hat{n}_A(v) - P(v \in A)| \leq \epsilon$. This follows by applying a multiplicative Chernoff bound for (1), Hoeffding's inequality for (2,3) and taking a union bound.

Assume that $\epsilon < 1/2$. With probability at least $1 - \frac{\delta}{2|V|^2}$:

$$
\begin{aligned}
\hat{L}(v \mid u) &\leq (P(v \in A, u \in S) + \epsilon)/((1-\epsilon)P(u \in S)) - P(v \in A) + \epsilon \\
&= (1-\epsilon)^{-1}[P(v \in A, u \in S) + \epsilon]/P(u \in S) - P(v \in A) + \epsilon \\
&= (1 + \frac{\epsilon}{1-\epsilon})[P(v \in A, u \in S) + \epsilon]/P(u \in S) - P(v \in A) + \epsilon \\
&\leq (1 + 2\epsilon)[P(v \in A, u \in S)/P(u \in S) + \epsilon/\Delta] - P(v \in A) + \epsilon \\
&\leq P(v \in A, u \in S)/P(u \in S) - P(v \in A) + 3\epsilon + (1+2\epsilon)\epsilon/\Delta \\
&\leq L(v \mid u) + 6\epsilon/\Delta
\end{aligned}
$$

The first inequality is a consequence of the concentration inequalities previously mentioned. The second inequality holds from assuming $\epsilon < 1/2$ and the fact that $P(u \in$

$S) = q_u > \Delta$. The third follows because $P(v \in A, u \in S)/P(u \in S) = P(v \in A \mid u \in A) < 1$. Taking $\epsilon = 1/12\Delta^2 L_{\min}$ gives $\hat{L}(v \mid u) \leq L(v \mid u) + \Delta L_{\min}/2$ with probability at least $1 - \frac{\delta}{2|V|^2}$. The symmetric argument lets us conclude the same for $\hat{L}(v \mid u) \geq L(v \mid u) - \Delta L_{\min}/2$. A union bound concludes the proof. □

4.4. Necessary Dependence on Minimum Lift

The only dependence of our algorithm whose necessity is not self-evident is that of the inverse minimum lift $\frac{1}{L_{\min}}$ in the sampling complexity bound of Theorem 3. We now show that this dependence is in fact unavoidable in the worst case: there are trees in which $L_{\min}$ can be arbitrarily small, and any algorithm for exact learning must sample approximately $\frac{1}{L_{\min}}$ observations.

The lower bound relies on the following construction. Let V consist of 4 "special" vertices $\{a, b, z_0, z_1\}$ and $2n-1$ additional vertices $V = \{z, x_1^{(0)}, ..., x_{n-1}^{(0)}, x_1^{(1)}, ..., x_{n-1}^{(1)}\}$.

We describe two trees. The tree T_a^n is given by first connecting z_0 to a and z_1 to b. T_b^n, on the other hand, has z_0 connected to b and z_1 connected to a. Everything else about both T_a^n and T_b^n will be the same. In particular, z_0 will connect to each of $x_i^{(0)}$, z_1 to each of $x_i^{(1)}$ and both z_0 and z_1 will be connected to z. Finally, let the infection probability be p for any edge, and seed probability be q for any vertex be Δ.

Now suppose that T^n is drawn from $\{T_a^n, T_b^n\}$ with equal probability. In order to exactly learn E, the learner must determine whether the edges $(a, z_0), (b, z_1)$ or the edges $(b, z_0), (a, z_1)$ exist.

The basic idea behind this construction is that the lift that either z_0 or z_1 provide to either a or b can be made arbitrarily small by making n large, since then z_0 and z_1 are infected with near certainty under a product distribution on seed sets. However, detecting whether we are in T_n^a or T_n^b requires distinguishing lifts between these four vertices. This leads to the following result, whose proof is technical and deferred to the Appendix.

Theorem 4. *For any ϵ, there exists an n such that $L_{\min}(T^n) < \epsilon$, and any algorithm must sample at least $(32\Delta L_{\min})^{-1}$ observations in order to reconstruct the edge set of T^n with probability at least $7/8$.*

5. A Generalized Lift-Based Algorithm

The algorithm described in Section 4 specifically leverages the fact that the underlying network is a tree. It computes all pairwise lifts in the network, then greedily selects the top $|V| - 1$ edges, as long as those edges do not create a cycle. Therefore, this algorithm will certainly fail to learn a network that is not a tree. However, even for non-trees, observe that the lifts $L(v \mid u)$ themselves remain well-defined. Moreover, we should continue to expect that lifts reveal relevant information about the structure of a network. A large lift $L(v \mid u)$ implies that that v is far more likely to be infected when u is seeded, relative to its background probability of being infected.

In this section we will consider a modified version of the lift algorithm more suitable to general networks. This algorithm (which we call *K-lifts*), expects as input estimates of lifts $\hat{L}(\cdot \mid \cdot)$, as before. It then simply greedily selects the K largest lifts, and constructs the edges between the vertices responsible for those lifts (see Algorithm 2).

Algorithm 2 K-lifts algorithm.

```
1: Input: Estimates $\hat{L}(\cdot \mid \cdot)$, vertex set $V$, $K > 0$.
2: Set $\hat{E} = \emptyset$
3: while $|\hat{E}| \neq K$ do
4:     Add $\arg\max_{\{u,v\} \notin \hat{E}} L(v \mid u)$ to $\hat{E}$.
5: end while
6: Return $(V, \hat{E})$
```

We will demonstrate that even when the network we are trying to learn is no longer a tree, this "lift-based" approach may still be effective. In other words, there is robustness to this approach even when the assumptions of Section 4 are violated. We first study the performance of the algorithm when learning Erdős-Reńyi networks. We discover that lifts continue to predict network structure under certain conditions. We then provide evidence that the algorithm can also perform well on more realistic networks.

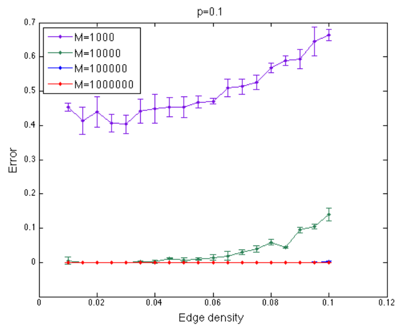
We conclude by proving some simple facts about the K-lifts algorithm. On the one hand, we show that the algorithm learns simple cycles. However, we also find a network which is sparse and nearly a tree on which the algorithm will make many mistakes. Thus, there are indeed networks which will provably foil the lift algorithm.

5.1. Experiments

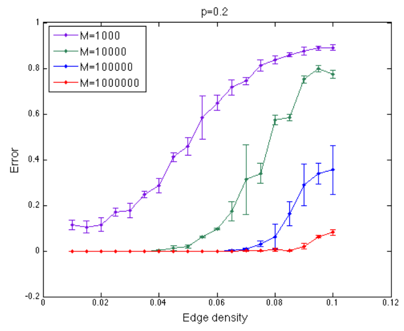
Let $G(n, \varphi)$ be an Erdős-Rényi random network[7] of size n, where each edge is present in the network independently with probability φ (which implies the edge density of the network is close to φ with high probability). For the underlying network in our experiments, we will consider $G(100, \varphi)$ for $\varphi = 0.01, ..., 0.1$. In all the experiments, each vertex becomes initially infected independently and with probability 0.05 (so roughly 5 vertices are seeded.)

Denote the number of edges in a network G by $m(G)$. For simplicity, we will set the algorithm's parameter $K =$

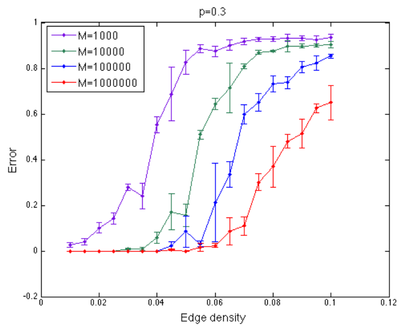
[7]Note that the traditional notation for this is $G(n, p)$, but we use p to indicate the transmission probability of an edge.



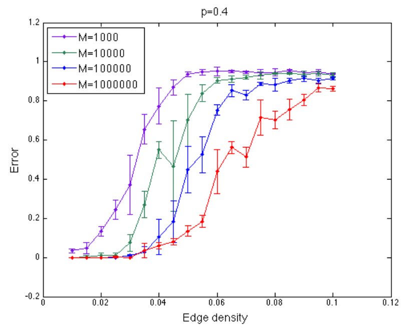
(a)



(b)



(c)



(d)

Figure 1. The edge density of the underlying network vs. the error rate of the K-lifts algorithm for different values of p. (The error bars around the points represent the 95% confidence interval; for each edge density φ, two random networks with edge density φ were generated, and the K-lift algorithm was run twice on each network.)

$m(G)$, the correct number of edges, in each experiment. In other words, the algorithm is told the correct number of edges. Since we are setting $K = m(G)$, it is meaningful to consider $\mathrm{err}(E, \hat{E}) = \frac{|\hat{E}\Delta E|}{m(G)}$, where E is the edge set of G and $\hat{E}$ is the edge set returned by the algorithm. In other words, $\mathrm{err}(E, \hat{E})$ is the fraction of those edges which the algorithm either fails to find or add mistakenly. For estimating the lifts, we need sufficient number of samples. Let M denote the number of samples available to the learning algorithm. We run the experiments for $M \in \{1000, 10000, 100000, 1000000\}$ and observe the relation between the performance of the K-lifts algorithm and the number of samples.

The results of the experiments are summarized in Figure 1. Each of these plots depicts the error of the K-lifts algorithm $\mathrm{err}(E, \hat{E})$ against the density of the network $D(G) = 2m(G)/(n(n-1))$, and each plot corresponds to a particular value of p. The experiments demonstrate that indeed the K-lifts algorithm enjoys some robustness even when the tree assumption is violated. This is most obvious in Figure 1 (a). However, this is not unconditional. There are two forces that make learning more difficult: increasing the edge density, or increasing the transmission probability p. Increasing either will cause the sample size required in order to attain a low error rate to increase in turn.

Why might this be the case? We hypothesize that the lift algorithm's performance begins to degrade when, with high probability, large portions of the network are infected after each contagion process. Alternatively, this is the point at which seeds no longer cause infections local to the seed's neighborhood. The lift algorithm fundamentally measures the difference in infection rate of a vertex u under two different circumstances. When any u tends to frequently be infected, more and more samples are required to detect this difference. As the network density φ and transmission probability p increase, at some point the probability of large-scale non-local infections becomes certain.

It is interesting to note that small p can hinder learning as well. After all, if $p = 0$, seed infections never spread, and there is nothing to learn. We see this phenomenon in Figure 1 (a). When $M = 1000$ and $p = 0.1$ we simply do not see enough infections to be able to accurately estimate lifts.

Notice also that since the algorithm is greedy, in the cases where it exactly learns the structure, the assumption $K = m(G)$ is mild. If a good estimate of $m(G)$ is provided instead, the algorithm will only make an additional number of mistakes equal to missing/extra edges in the estimate.

Other networks. We also ran the generalized lift algorithm on more realistic models and data sets, including a real collaboration network, NetScience (Newman, 2003; Boccaletti et al., 2006; Newman, 2006), and networks generated by the Small Worlds model (Watts & Strogatz, 1998).

On the real collaboration network, NetScience, which consists of 1589 vertices and 2742 edges, we continue to observe low reconstruction error. We obtain an error rate near 0 when edges have an infection transmission probability of 10%, an error rate of 0.06 when the transmission probability is 20%, and an error rate of 0.16 when the transmission probability is 30%.

On networks generated from the Small Worlds model, reconstruction error ranges from 0 to a bit above 0.1 at transmission probability 20% or smaller. As perhaps expected, performance improves as we increase the amount of rewiring in the Small Worlds model, moving from networks with very high clustering and many short cycles towards random connectivity. At transmission probability 30% the degradation of the algorithm is more rapid. Results are similar but slightly better for networks generated from a less symmetric model that also balances high clustering with low diameter.

5.2. Theoretical Robustness and Limitations

We conclude with theoretical results that partially characterize the behavior of the K-lift algorithm on cyclical networks. To prove these results we will need a utility lemma that helps us characterize the lift between two vertices. Let $X(u)$ be the active component of u (see Section 4.1).

Lemma 5. *On an arbitrary network G, $L(v \mid u) = P(v \in X(u), X(u)$ is not infected$) = P(u \in X(v), X(v)$ is not infected$)$.*

Proof. The proof follows along identically to the proof of Lemma 3 by replacing the event $\mathcal{C}(u,v)$ with $\{v \in X(u)\}$, and omitting the final substitutions for ϕ, ψ. $\square$

We can now show that the K-lift algorithm exactly learns simple cycles.

Theorem 5. *Let $C = (V, E)$ be a cycle on $n > 5$ vertices and $p_{(u,v)} = p \leq \frac{1}{2}$, $q_u = q$ for all u, v. There exists a $\Delta > 0$ such that any $(u,v) \in E$ and $(w,z) \notin E$, satisfy $L(v \mid u) - L(w \mid z) \geq \Delta$.*

Proof. By symmetry, $L(v \mid u)$ is the same for any edge (u,v) in E and $L(w \mid z)$ is the same for any w and z distance d apart. So without loss of generality, we show that $L(v \mid u) - L(w \mid u) \geq \Delta$ where v is immediately clockwise to u and w is of distance $d \geq 2$ clockwise to u.

Let E^+ be the edges between v and w moving clockwise, and φ be the probability that one of these edges is inactive. Let E^- be the edges between u and w moving counterclockwise, and V^- be the edges between u and w moving counterclockwise, inclusive. Define A to be the event that $v \in X(u)$, $w \notin X(u)$ and $X(u)$ is not infected. Define B be the event that $w \in X(u)$, $v \notin X(u)$ and $X(u)$ is not infected. And Define F the event that $w, v \in X(u)$ and $X(u)$ is not infected. Lemma 5 and the fact that these events are disjoint tells us that $L(v \mid u) - L(w \mid u) = P(A) + P(F) - P(B) - P(F) = P(A) - P(B)$. However B can only occur if (u,v) is inactive, every edge in E^- is active, and some edge in E^+ is inactive. Furthermore B requires that all of V^- be unseeded. Thus, $P(B) \leq (1-p)p^{n-d}\varphi(1-q)^{n-d+1}$. Since $d \leq \frac{n}{2}$ and $\varphi \leq 1$, $P(B) \leq (1-p)p^{n/2}\varphi(1-q)^{n/2+1} \leq (1-p)p^{n/2}(1-q)^{n/2+1}$. Also, A occurs if (u,v) is active, the edge immediately clockwise to v and the edge immediately counterclockwise of u are inactive, and u and v are not seeded. Thus, $P(A) \geq p(1-p)^2(1-q)^2$. The fact that $n > 5$ and $p \leq \frac{1}{2}$, proves the theorem. $\square$

While this may seem promising, there exist networks that are almost trees – trees with a single edge added – on which the K-lift algorithm would make many mistakes. Consider the following network on $2n-1$ vertices: Let S be a star consisting of $n-1$ vertices centered at v_0. Let C be a cycle consisting of the remaining n vertices as well as v_0. Define $G = S \cup C$. We use the notation $V(C)$ to refer to the vertex set of C, with $E(C), V(S), E(S)$ defined analogously.

Theorem 6. *Suppose $p_{(u,v)} = q_u = 1/2$ for all u, v. When run on G with $K = m(G) = 2n-1$, the K-lifts algorithm returns $\hat{E}$ satisfying $\mathrm{err}(E, \hat{E}) > 1/2 - o(1)$.*

Proof. Name the vertices in C, $v_0, v_1, ..., v_n$ according to the order in which they appear in the cycle. Let U consist of all pairs of vertices on C which are distance 2 apart without crossing v_0. More explicitly $U = (v_1, v_3), (v_2, v_4), ..., (v_{n-2}, v_n)$. It suffices to show that for any $(v, v_0) \in E(S)$, and any $(x,y) \in U$, $L(x \mid y) > L(v \mid v_0)$, which means, the K-lift algorithm will build the chords in U before it considers the edges in the star S, making $n-2$ mistakes. By Lemma 5, $L(v \mid v_0) < P(v_0 \notin A) < (1-qp)^{n-1}$. On the other hand, for any $(x,y) \in U$ the probability that x and y belong to the same uninfected component is at least $p^2(1-p)^2(1-q)^4$. Therefore Lemma 5 also implies that $L(x \mid y) \geq p^2(1-p)^2(1-q)^4$. A sufficiently large n proves the theorem. $\square$

Acknowledgement. We give warm thanks to Moez Draief for discussions on the results presented here.

References

Abrahao, Bruno, Chierichetti, Flavio, Kleinberg, Robert, and Panconesi, Alessandro. Trace complexity of network inference. In *KDD*, 2012.

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D.-U. Complex networks: Structure and dynamics. *Physics Reports*, (424), 2006.

Du, Nan, Song, Le, Smola, Alexander J, and Yuan, Ming. Learning networks of heterogeneous influence. In *NIPS*, 2012.

Goldenberg, Jacob, Libai, Barak, and Muller, Eitan. Talk of the network: A complex systems look at the underlying process of word-of-mouth. In *Marketing Letters, 123: 211-223*, 2001a.

Goldenberg, Jacob, Libai, Barak, and Muller, Eitan. Using complex systems analysis to advance marketing theory development. In *Academy of Marketing Science Review*, 2001b.

Gomez-Rodriguez, Manuel, Leskovec, Jure, and Krause, Andreas. Inferring networks of diffusion and influence. In *KDD*, 2010.

Gripon, Vincent and Rabbat, Michael. Reconstructing a graph from path traces. In *CoRR abs/13016916*, 2013.

Kempe, David, Kleinberg, Jon, and Tardos, Eva. Maximizing the spread of influence through a social network. In *KDD*, 2003.

Lippert, Christoph, Stegle, Oliver, Ghahramani, Zhoubin, and Borgwardt, Karsten M. A kernel method for unsupervised structured network inference. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics AISTATS*, 2009.

Myers, Seth A and Leskovec, Jure. On the convexity of latent social network inference. In *NIPS*, 2010.

Netrapalli, Praneeth and Sanghavi, Sujay. Learning the graph of epidemic cascades. In *SIGMETRICS*, 2012.

Newman, Mark. The structure and function of complex networks. *SIAM Review*, (45), 2003.

Newman, Mark. Finding community structure in networks using the eigenvectors of matrices. *Preprint*, 2006.

Rodriguez, Manuel and lkopf, Bernhard. Submodular inference of diffusion networks from multiple trees. In *ICML*, 2012.

Rodriguez, Manuel Gomez, Balduzzi, David, and Schlkopf, Bernhard. Uncovering the temporal dynamics of diffusion networks. In *ICML*, 2011.

Vert, Jean-Philippe and Yamanishi, Yoshihiro. Supervised graph inference. In *NIPS*, 2005.

Watts, Duncan and Strogatz, Steven. Collective dynamics of small-world networks. *Nature*, (393), June 1998.