STRUCTURE AND DYNAMICS OF DIFFUSION NETWORKS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL
ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Manuel Gomez Rodriguez
May 2013

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Andrew Ng, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Emmanuel Candes

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jurij Leskovec

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Bernhard Scholkopf

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

Abstract

Diffusion of information, ideas, behaviors and diseases are ubiquitous in nature and modern society. One of the main goals of this dissertation is to shed light on the hidden underlying structure of diffusion. To this aim, we developed flexible probabilistic models and inference algorithms that make minimal assumptions about the physical, biological or cognitive mechanisms responsible for diffusion. We avoid modeling the mechanisms underlying individual activations, and instead develop a data-driven approach which uses only the visible temporal traces diffusion generates.

We first developed two algorithms, NetInf and Multitree, that infer the network structure or *skeleton* over which diffusion takes place. However, both algorithms assume networks to be static and diffusion to occur at equal *rates* across different edges. We then developed NetRate, an algorithm that allows for static and dynamic networks with different *rates* across different edges. NetRate infers not only the network structure but also the *rate* of every edge. Finally, we develop a general theoretical framework of diffusion based on survival theory.

Our models and algorithms provide computational lenses for understanding the structure and temporal dynamics that govern diffusion and may help towards forecasting, influencing and retarding diffusion, broadly construed. As an application, we study information propagation in the online media space. We find that the information network of media sites and blogs tends to have a core-periphery structure with a small set of core media sites that diffuse information to the rest of the Web. These sites tend to have stable circles of influence with more general news media sites acting as connectors between them. Information pathways for general recurrent topics are more stable across time than for on-going news events. Clusters of news media sites and blogs often emerge and vanish in matter of days for on-going news events. Major social movements and events involving civil population, such as the Libyan's civil war or Syria's uprise, lead to an increased amount of information pathways among blogs as well as in the overall increase in the network centrality of blogs and social media sites.

Additionally, we apply our probabilistic framework of diffusion to the influence maximization problem and develop the algorithm InfluMax. Experiments on synthetic and real diffusion networks show that our algorithm outperforms other state of the art algorithms by considering the temporal dynamics of diffusion.

Acknowledgement

I would like to thank my advisor Andrew Ng for his advice and support. Hopping between Stanford University and MPI for Intelligent Systems would not have been possible without his help.

I am deeply grateful to my co-advisor Bernhard Schölkopf, who I consider truly a *Doktorvater*, for his mentorship, advice and support. I have learned a lot from his views on what is research all about, and the freedom and encouragement he has given me over the years have been exceptional.

I would also like to thank my first reader Jure Leskovec for introducing me to research on networks. I truly admire his drive and passion. He has always managed to challenge me, and has been a great source of motivation and advice. It has been an honor to count with Emmanuel Candes as the second reader for this dissertation, and I would like to thank him for his advice and comments. I would also like to thank Christopher Potts for accepting the invitation to chair my orals committee.

I would also like to thank my coauthors Andreas Krause, David Balduzzi, Jan Peters, Moritz Grosse-Wentrup, Jeremy Hill, Alireza Gharabaghi, Monica Rogati, Georgios Naros, and Jens Kober. I would like to thank Jan and Moritz for their mentoring and advice and colleagues from MPI for Intelligent Systems for great scientific discussions.

I would like to thank my friends from Stanford (Borja, Gemma, Yiannis, Sotiria, Nicholas, Hylke, Ana, Amir, ...) for inspiring conversations, dinners, and trips. I would also like to thank my friends from Tübingen, specially my officemate Michel, for great endless conversations inside and outside the office, Tim, for his Californian optimism and his animal style American burgers, Shih-pi, for her support and unconditional friendship, Ander, for keeping me a little bit fitter and always trying to keep me out of the office, and Lana, for her love, for helping me having a more balanced life, including becoming a tango dancer, and for being an adventurous traveler. I have great memories from my year in Limerick and would like to thank specially Karen, Alan, Gabi and Xavi, for making me having a great time despite of the weather. I am also very fortunate to have still many good friends in Pontevedra, my hometown, despite not spending much time there anymore: Yoli, both Miguis, Santi, Gon, Oscar, Robert, and many others, who always welcome me back as if time wouldn't have passed.

Last but not least, I am very grateful to my parents Enrique and Margarita, for their unconditional love, kindness and values and to my sister Maria for always believing in me and my skills, even when I did not. Os dedico esta tesis a los tres!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many interacting systems can be fruitfully recast in terms of signals propagating over networks. In recent years, there has been an increasing effort to uncover, understand, and influence a broad range of propagation processes arising over a wide variety of network structures: information propagation (Katz and Lazarsfeld, 1955; Adar and Adamic, 2005), social networks (Kempe et al., 2003; Lappas et al., 2010; Mathioudakis et al., 2011), viral marketing (Domingos and Richardson, 2001; Watts and Dodds, 2007; Aral and Walker, 2012), technical innovations (Rogers, 1995), computer viruses (Wang et al., 2000), human travel (Brockmann et al., 2006) and epidemiology (Lipsitch et al., 2003; Hufnagel et al., 2004; Wallinga and Teunis, 2004).

Abstractly, we think of a *contagion* that appears at some node of a network and then spreads like an epidemic from node to node over the edges of the network. For example, in information propagation, the contagion corresponds to a piece of information (Liben-Nowell and Kleinberg, 2008; Leskovec et al., 2009), the nodes correspond to people and infection events are the times when nodes learn about the information. Similarly, we can think about the spread of a new type of behavior or an action, e.g., purchasing a new product (Leskovec et al., 2006a), or the propagation of a contagious disease over social network of individuals (Bailey, 1975).

In this context, many research problems have been raised in the last years, ranging from network inference based on diffusion traces (Gomez-Rodriguez et al., 2010, 2011), finding culprits or end effectors from diffusion traces (Lappas et al., 2010;

Shah and Zaman, 2011; Prakash et al., 2012a) and reconstructing incomplete diffusion traces (Chierichetti et al., 2011; Sadikov et al., 2011) to influence spread minimization (Budak et al., 2011; Blume et al., 2011) and maximization (Kempe et al., 2003; Aral and Walker, 2012).

This dissertation is mainly devoted to one of the fundamental research problems in the context of network diffusion: inference of hidden or implicit networks over which various types of *contagions* spread (Gomez-Rodriguez et al., 2010, 2011, 2012; Gomez-Rodriguez and Schölkopf, 2012b,c; Gomez-Rodriguez et al., 2013). Additionally, it also tackles the influence (spread) maximization over networks (Gomez-Rodriguez and Schölkopf, 2012a) from a novel perspective, in which influence or information can spread at different rates across different edges, as in real-world examples.

1.1 Inference of diffusion networks

Observing a diffusion process often reduces to noting when nodes (people, blogs, etc.) reproduce a piece of information, get infected by a virus, or buy a product. Epidemiologists can observe when a person becomes ill but they cannot tell who infected her or how many exposures and how much time was necessary for the infection to take hold. In information propagation, we observe when a blog mentions a piece of information. However if, as is often the case, the blogger does not link to her source, we do not know where she acquired the information or how long it took her to post it. Finally, viral marketers can track when customers buy products or subscribe to services, but typically cannot observe who influenced customers' decisions, how long they took to make up their minds, or when they passed recommendations on to other customers. In all these scenarios, we observe *where and when* but *not how or why* information (be it in the form of a virus, a meme, or a decision) propagates through a population of individuals. The mechanism underlying the process is hidden. However, the mechanism is of outstanding interest in all three cases, since understanding diffusion is necessary for stopping infections, predicting meme propagation, or maximizing sales of a product.

To the best of our knowledge, we were among the first to study the inference of diffusion networks from diffusion traces or *cascade* data, and to develop an efficient network inference algorithm, NetInf (Gomez-Rodriguez et al., 2010, 2012). This work received a best research paper award honorable mention at ACM KDD 2010 and by March 2013, accumulated more than 170 citations, according to Google Scholar. We devote Chapter 3 of this dissertation to this first approach and a follow-up algorithm, Multitree (Gomez-Rodriguez and Schölkopf, 2012c). Both algorithms assume that networks do not change over time and diffusion along different edges of a network occur at the same transmission rate. In other words, they assume networks to be *static* and *unweighted*. Both NetInf and Multitree are scalable approximation algorithms with provable near-optimal performance based on submodular maximization. We apply both algorithms to information diffusion among mainstream media and blogs sites and experiment with more than 170 million blogs and news articles. We find that the diffusion network of news for the top 1,000 media sites and blogs tends to have a core-periphery structure with a small set of core media sites that diffuse information to the rest of the Web. These sites tend to have stable circles of influence with more general news media sites acting as connectors between them.

In Chapter 4, we consider *weighted* networks, where diffusion occurs at different rates across different edges so that we can infer temporally heterogeneous interactions within a network, as found in real-world examples, by developing NetRate (Gomez-Rodriguez et al., 2011) and InfoPath (Gomez-Rodriguez et al., 2013). Both methods allow for *static* and *dynamic* networks that change over time, depending upon the contagions that propagate through them (Myers et al., 2012; Romero et al., 2011a). This is important since, for example, a blog can increase its popularity abruptly after one of its posts turns *viral*, this may create new edges in the information transmission network and so the content the blog produces in the future will likely spread to larger parts of the network. Similarly, at any given time a particular unexpected event may occur and a topic or piece of news may become very popular for a limited period of time. This again will lead to different emerging and vanishing information pathways, and thus to a dynamic underlying network. Both algorithms involve convex programming, naturally (without heuristics) imposes sparse solutions and requires no parameter tuning. We apply both algorithms to information diffusion among mainstream media and blog sites and experiment with more than 170 million different pieces of information spreading over the network in a one year period. We find that information pathways for general recurrent topics are more stable across time than for on-going news events. Clusters of news media sites and blogs often emerge and vanish in matter of days for on-going news events. Major social movements and events involving civil population, such as the Libyan's civil war or Syria's uprise, lead to an increased amount of information pathways among blogs as well as in the overall increase in the network centrality of blogs and social media sites.

Finally, Chapter 6 presents a general theoretical framework for studying information propagation over networks based on survival theory, which allow us to generalize previous methods.

1.2 Influence maximization in diffusion networks

Influence spread maximization tackles the problem of selecting the most influential source node set of a given size in a diffusion network. A diffusion process that starts in such an influential set of nodes is expected to reach the greatest number of nodes in the network. In information propagation, the problem reduces to choosing the set of blogs and news media sites that together are expected to spread a piece of news to the greatest number of sites. In viral marketing, it consists of identifying the most influential set of *trendsetters* that together may influence the greatest number of customers. Finally, in epidemiology, the influence maximization problem reduces to finding the set of individuals that together are most likely to spread an illness or virus to the greatest percentage of the population. In this latter case, the solution of the influence maximization problem would help with developing vaccination and quarantine policies.

In Chapter 5, we build on the continuous time model of diffusion which we originally developed to tackle the network inference problem over weighted networks, described in Chapter 5. As noted in the previous section, this model accounts for temporally heterogeneous interactions within a diffusion network – it allows information (or influence) to spread at different rates across different edges, as shown in real-world examples. We develop a method for influence maximization, InfluMax, that accounts for the temporal dynamics underlying diffusion processes. The method evaluates influence analytically using continuous time Markov chains (CTMCs) and finds a suboptimal set of source nodes with *provable guarantees* in terms of the average influence.

To the best of our knowledge, previous work on influence maximization has ignored the underlying temporal dynamics governing diffusion networks – once a transmission occurs, it always occurs at the same rate or temporal scale (Richardson and Domingos, 2002; Kempe et al., 2003; Bharathi et al., 2007; Carnes et al., 2007; Chen et al., 2009, 2010; Goyal et al., 2010b; Budak et al., 2011; Chen et al., 2011; Irfan and Ortiz, 2011). In contrast, we consider heterogeneous pairwise transmission rates since this is a key point in many real-world examples. In information propagation, news media sites and professional bloggers typically report news faster than people that maintain personal blogs. In epidemiology, people meet each other with different frequencies and then the pairwise transmission rates between individuals within a population differ. Finally, in viral marketing, some customers make up their minds about a product or service quicker than others, and then pass recommendations on to other customers at different rates.

The main contribution of our work on influence maximization is twofold. First, it considers a novel continuous time formulation of the influence maximization problem in which information or influence can spread at different rates across different edges, as in real-world examples. Second, this continuous time approach allows us to analytically compute and optimize the influence (*i.e.*, average total number of infections) using CTMCs and submodular maximization, avoiding the use of heuristics (Chen et al., 2010, 2009) or Monte Carlo simulations (Kempe et al., 2003).

1.3 Thesis organization

The remainder of this dissertation is organized as follows. Chapter 2 provides relevant background and concepts used throughout the thesis. Chapter 3 is devoted to inferring unweighted static diffusion networks from cascade data, and presents two inference algorithms based on submodular optimization: NetInf and Multitree. Chapter 4 introduces the problem of inferring weighted static and dynamic diffusion networks from cascade data, and presents an inference algorithm based on convex optimization: NetRate. Chapter 5 tackles the problem of maximizing the influence in a diffusion network, and presents an influence maximization algorithm based on Continuous Time Markov Chains (CTMC) and submodularity: InfluMax. In Chapter 6, we propose a general theoretical framework to model propagation and infer hidden or unobserved networks using survival theory. Finally, Chapter 7 concludes this dissertation.

This dissertation covers material from the following publications:

1. M. Gomez-Rodriguez, J. Leskovec and B. Schölkopf. Modeling Information Propagation with Survival Theory. In *ICML '13: Proceedings of the 30th International Conference on Machine Learning*, 2013.

2. M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf. Structure and Dynamics of Information Pathways in Online Media. In *WSDM '13: Proceedings of the 6th International Conference on Web Search and Data Mining*, 2013.

3. M. Gomez-Rodriguez and B. Schölkopf. Modeling Information Propagation with Survival Theory. In *NIPS '12: Advances in Neural Information Processing Systems: Workshop in Algorithmic and Statistical Approaches for Large Social Networks*, 2012.

4. M. Gomez-Rodriguez and B. Schölkopf. Influence Maximization in Continuous Time Diffusion Networks. In *ICML '12: Proceedings of the 29th International Conference on Machine Learning*, 2012.

5. M. Gomez-Rodriguez and B. Schölkopf. Submodular Inference of Diffusion Networks from Multiple Trees. In *ICML '12: Proceedings of the 29th International Conference on Machine Learning*, 2012.

6. M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD)*,

Volume 5, Number 4, 2012.

7. M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the Temporal Dynamics of Diffusion Networks. In *ICML '11: Proceedings of the 28th International Conference on Machine Learning*, 2011.

8. M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring Networks of Diffusion and Influence. In *KDD '10: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010 (**Best Research Paper Award Honorable Mention**)

Chapter 2

Background and basic concepts

2.1 General network-theoretic concepts

In this section, we briefly revisit basic graph-theoretic concepts that we will use throughout this dissertation.

Network. A network (or graph) is an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ comprising a set $\mathcal{V}$ of nodes or vertices together with a set $\mathcal{E}$ of edges, links or lines, which are 2-element subsets of $\mathcal{V}$ (*i.e.*, an edge is related with two vertices). In the remainder of this thesis, we will use network and graph indistinctly.

Undirected and directed networks. An undirected network is a network in which edges have no orientation, *i.e.*, the edges are unordered pairs of nodes. In contrast, in a directed network, edges have orientation, *i.e.*, $(i, j) \neq (j, i)$, and for every node, we can distinguish out-edges (out-links) and in-edges (in-links).

Node degree. The degree of a node of a network is the number of edges incident to the node. For directed networks, we distinguish between in-degree and out-degree of a node. In-degree is the number of edges pointing towards the node, and out-degree is the number of edges pointing from the node.

Path. A path in a network is a sequence of edges which connect a sequence of nodes. For directed networks, we distinguish between undirected and directed paths. In a directed path, all edges go into the same direction, and in a undirected path, the directions may differ.

Weakly and strongly connected networks. A directed network is weakly connected if there exists an undirected path connecting any pair of nodes, and is strongly connected is there exists a directed path connecting any pair of nodes in network.

Diameter. Network $\mathcal{G}$ has a diameter d if the maximum length of the undirected shortest paths over all connected pairs of nodes is d. The length of the path is the number of edges it contains.

Community. Perhaps surprisingly, there is not a real consensus on the definition of community. However, a community is often defined as a set of nodes in a network $\mathcal{G}$ that has more and/or *better*-connected edges between its members than between members of the set and the remainder of the network. This definition matches the traditional definition of cluster.

Complete network. An undirected network is complete if all pairs of nodes are connected. A directed network is complete if all pairs of nodes are connected by a pair of edges, one in each direction.

Empty network. A network is empty if it has no edges.

Subnetwork. A subnetwork $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ of a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a network whose edge set is a subset of that of $\mathcal{G}$ and its vertex set is a subset of that of $\mathcal{G}$ restricted to the endpoints in the edge subset: $\mathcal{E}_s \subseteq \mathcal{E}$ and $\mathcal{V}_s = \{i, j : (i, j) \in \mathcal{E}_s\}$.

Induced subnetwork. An induced (or full) subnetwork $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a network whose vertex set is a subset of that of $\mathcal{G}$, and its vertex set is composed of all edges in $\mathcal{G}$ between nodes in the vertex subset: $\mathcal{V}_s \subseteq \mathcal{V}$ and $\mathcal{E}_s = \{(i, j) : (i, j) \in \mathcal{E} \wedge i, j \in \mathcal{V}_s\}$.

Directed tree. A directed tree $\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$ is a directed network in which, for a node u called the root and any other node v, there is exactly one directed path from u to v. Our definition of directed tree is equivalent to the standard definition of arborescence.

Directed spanning tree. A directed spanning tree $\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$ of a directed network $G = (\mathcal{V}, \mathcal{E})$ is a directed tree that contains all nodes of G: $\mathcal{V}_\mathcal{T} = \mathcal{V}$ and

$\mathcal{E}_{\mathcal{T}} \subseteq \mathcal{E}$.

Directed acyclic network. A directed acyclic network (or directed acyclic graph, DAG) is a directed network with no directed cycles. In other words, there is no way to start at some node v and follow a directed path of edges that eventually loops back to v again.

Dominated node set. Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a set of nodes $\mathcal{B} \subseteq \mathcal{V}$, and a node $n \in \mathcal{V}$, we define the dominated node set: $S_n(\mathcal{B}) = \{u \in \mathcal{V} :$ any directed path from u to n in G visits at least one node in $\mathcal{B}\}$.

Self dominant node set. Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a node $n \in \mathcal{V}$, a set $\mathcal{B}$ is self dominant if and only if $S_n(\mathcal{B}) = \mathcal{B}$ or, equivalently, if and only if the node set dominated by $\mathcal{B}$ is $\mathcal{B}$ itself.

2.2 Models of network structure

Erdős and Rényi (1960) were among the first to introduce a probabilistic generative model for (undirected) networks. In their random graph model, each pair of nodes has an i.i.d. probability of being connected by an edge. Since then, there has been a great amount of work on the theoretical properties of their model. Unfortunately, their model is not able to generate network properties that appear in real-world networks such as heavy-tailed in-degrees and out-degrees, communities (or clusters), densification power-law, and shrinking diameter.

Many recent network models build on the idea of preferential attachment (Barabási and Albert, 1999; Albert and Barabási, 2002; Winick and Jamin, 2002; Kleinberg et al., 1999; Kumar et al., 2000; Flaxman et al., 2006): a network is generated greedily and at each iteration, a new node u joins the network, and creates a link to an existing node v with the probability proportional to the degree of the node v. This leads to a "rich get richer" phenomena, and heavy-tailed degree distributions, but not necessarily to communities, densification power-law and shrinking diameter. In order to overcome this limitation, several variations of preferential attachment have been proposed: the copying model (Kleinberg et al., 1999; Kumar et al., 2000), the "winner does not take all model Pennock et al. (2002), the Community Guided Attachment model (CGA) (Leskovec et al., 2005), the random surfer model Blum et al. (2006) and the Forest Fire model Leskovec et al. (2005). Interestingly, the latter is able to generate networks with heavy-tailed in-degrees and out-degrees, communities (or clusters), densification power-law, and shrinking diameter.

However, there are also models that do not rely on preferential attachment, such as the small-world model (Watts and Strogatz, 1998) and the Waxman generator (Waxman, 1988), which result in networks with small diameter and local clustering, and the Kronecker Graph model (Leskovec et al., 2010), which are able to generate networks with heavy-tailed in-degrees and out-degrees, communities (or clusters), densification power-law, and shrinking diameter. Importantly, the Kronecker Graph model is analytically tractable and allows for a rigorous analysis, in contrast with models that build on the idea of preferential attachment.

Throughout this thesis, we consider two models of directed real-world social and information networks: the Forest Fire (scale free) model and the Kronecker Graph model, which we briefly introduce next.

2.2.1 Forest Fire model

In this section, we provide a brief introduction to the Forest Fire model, and we refer the reader to Leskovec et al. (2005) for a more extensive overview.

The Forest Fire model combines the intuition behind several older models, such as the Community Guided Attachment model (CGA) and the copying model, to produce synthetic networks that exhibit several properties that has been observed in real networks. In particular, they satisfy the network properties mentioned above: heavy-tailed in-degrees and out-degrees, communities (or clusters), densification power-law, and shrinking diameter.

In the basic Forest Fire model, we need to set two parameters: a forward burning probability p and a backward burning ratio r, whose roles we describe below. The model generates a network greedily. The process starts with a single node network $\mathcal{G}_1$. Then, at every iteration t, it adds a new node v to the network $\mathcal{G}_t$ and creates its out-links as follows:

1. Node v first chooses an *ambassador* node w uniformly at random and forms a link to w.

2. Two random numbers, x and y, are drawn from geometric distributions with means $p/(1-p)$ and $rp/(1-rp)$, respectively. Node v selects x out-links and y in-links of w which are incident to nodes that have not yet been visited. Let $w_1, w_2, \ldots, w_{x+y}$ denotes the other ends of these selected links. If there are not enough in- or out-links in the current network, v selects as many as it can.

3. We apply step (2) recursively to each of $w_1, w_2, \ldots, w_{x+y}$. As the process continues, nodes cannot be visited a second time, preventing the construction from cycling.

A rigorous analysis of the Forest Fire model appears to be quite difficult. However, it has been shown experimentally to result in the desirable network properties mentioned above (Leskovec et al., 2005). Interestingly, depending on the forward and backward burning parameters, p and r, the model is capable of generating sparse or dense networks with diameters that either increase or decrease, while also producing heavy-tailed in- and out-degree distributions.

Throughout this dissertation, we use a more flexible extension of the Forest Fire model which includes *orphans* and multiple *ambassadors*. Orphans are nodes with no links. Nodes with multiple ambassadors can choose more than one ambassador with some positive probability when generating their out- and in-links.

2.2.2 Kronecker Graph model

In this section, we provide a brief introduction to the Kronecker Graph model, and we refer the reader to Leskovec et al. (2010) for a more extensive overview.

The Kronecker Graph model matches multiple properties of real networks: heavy-tailed in-degrees and out-degrees, communities (or clusters), densification power-law, and shrinking diameter, but at the same time it is analytically tractable and allows for a rigorous analysis, in contrast with the Forest Fire model. The main intuition behind the model is to create self-similar graphs, recursively. We begin with an initiator graph K_1, with N_1 nodes and E_1 edges, and by recursion we produce successively larger graphs $K_2, K_3, \ldots$ such that $N_k = N_1^k$ and $E_k = E_1^k$. The Kronecker Graph model achieves this by using Kronecker products between matrices.

Definition 1 (Kronecker product of matrices). *Given two matrices* $\mathbf{A}^{n \times m}$ *and* $\mathbf{B}^{n' \times m'}$, *the Kronecker product matrix* $\mathbf{C}^{n \cdot n' \times m \cdot m'}$ *is given by:*

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} \doteq \begin{pmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \ldots & a_{1,m}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,1}\mathbf{B} & \ldots & a_{2,m}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}\mathbf{B} & a_{n,1}\mathbf{B} & \ldots & a_{n,m}\mathbf{B} \end{pmatrix}$$

The deterministic Kronecker Graph model proposes to produce a network by building a growing sequence of matrices using the Kronecker product:

Definition 2 (Kronecker power). *The* k^{th} *power of* $\mathbf{K}_1$ *is defined as:*

$$\mathbf{K}_1^{[k]} = \mathbf{K}_k = \underbrace{\mathbf{K}_1 \otimes \mathbf{K}_1 \otimes \ldots \otimes \mathbf{K}_1}_{k \ times} = \mathbf{K}_{k-1} \otimes \mathbf{K}_1$$

Definition 3 (Kronecker network). *A Kronecker network of order* k *is defined by the adjacency matrix* $\mathbf{K}_1^{[k]}$, *where* $\mathbf{K}_1$ *is the Kronecker initiator adjacency matrix.*

For a number of reasons, discussed in Leskovec et al. (2010), it is desirable to include stochasticity in the process. To this aim, the entries of the initiator matrix are allowed to take values on the interval $[0, 1]$ and each entry encodes the probability of that particular edge appearing. Importantly, after applying the Kronecker power to such initiator matrix, we obtain a larger stochastic adjacency matrix, where again each entry of the large matrix gives the probability of that particular edge appearing in a larger network. Such a stochastic adjacency matrix denes a probability distribution over all networks. To obtain a network, an instance from this distribution by sampling individual edges is drawn. This leads to the following definition:

Definition 4 (Stochastic Kronecker network). *A Kronecker network of order* k

is defined by the adjacency matrix $\mathbf{K}_1^{[k]}$*, where* $\mathbf{K}_1$ *is the Kronecker initiator adjacency matrix.*

Throughout this dissertation, we use the stochastic Kronecker network model. Importantly, by varying the initiator matrix, we obtain different network topologies:

- Random networks (Erdős and Rényi, 1960), with initiator matrix $[0.5, 0.5; 0.5, 0.5]$.
- Core-Periphery networks (Leskovec et al., 2008), with initiator matrix $[0.9, 0.5; 0.5, 0.1]$.
- Networks with hierarchical community structure (Clauset et al., 2008), with initiator matrix $[0.9, 0.1; 0.1, 0.9]$

2.3 General diffusion-theoretic concepts

In this section, we introduce basic diffusion-theoretic concepts that we will use throughout this dissertation.

Contagion. Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a contagion c is the item (be it in the form of a disease, a piece of information, an idea or a behavior) which propagates through the nodes in the network $\mathcal{G}$. Any node $u \in \mathcal{G}$ may get activated, infected, or hit by the contagion c at an activation (infection) time t_u^c. In an information propagation setting, the activation time of a node (website) is simply the time when the node first heard of or mentioned the piece of information. In epidemiology, it is the time when the node (person) got infected by a virus or in marketing, it is the time when the node (customer) purchased a product.

Given a directed network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a contagion c, we define:

Source set and source node. The source set $\mathcal{S}_c \in \mathcal{G}$ is the set of nodes that first got activated by the contagion c. We assume that all nodes in the source set got activated at the same activation time t_0^c. The source set may be composed of a single source node.

Observation window. Given the activation time t_0^c of the source(s) for the contagion c, we define the observation window $[t_0^c, t_0^c + T^c]$ as the time interval during which node activations are recorded. T^c is called the observation window cut-off or time horizon.

Transmission time. Assume node i got activated at time t_i^c and then activates node j at time t_j^c. The transmission time $(t_j^c - t_i^c)$ quantifies how long it took for the contagion to spread from node i to node j.

Cascade. Given an observation window $[t_0^c, t_0^c + T^c]$, a cascade $\mathbf{t}^c = (t_1^c, \ldots, t_{|\mathcal{V}|}^c)$ is a $|\mathcal{V}|$-dimensional vector recording when each of the nodes in the network $\mathcal{G}$ got activated during the observation window. Symbol ∞ labels nodes that are not activated by the contagion c during observation window – it does not imply that nodes are never activated, except if $T^c \to \infty$.

Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define:

Transmission rate. The transmission rate $\alpha_{i,j}$ of an edge $(i, j) \in \mathcal{E}$ quantifies how frequently any contagion spreads from node i to node j or, in other words, the *latency* of the edge (i, j). We consider the transmission rate of every edge of the network $\mathcal{G}$ to be positive and transmission rates between disconnected nodes to be zero: $\alpha_{i,j} > 0$ if $(i, j) \in \mathcal{E}$, otherwise $\alpha_{i,j} = 0$.

Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a matrix of transmission rates $\mathbf{A} \doteq [\alpha_{i,j}]$ such that $\alpha_{i,j} > 0$ if $(i, j) \in \mathcal{E}$, we define:

Diffusion network. A diffusion network is the pair $(\mathcal{G}, \mathbf{A})$. In a diffusion network, a contagion appears at some node(s) of the network and then spreads from node to node over the edges of the network, following directed paths. In other words, a diffusion network is a 'skeleton' for the spread of information, ideas, behaviors and diseases.

Unweighted diffusion network. An unweighted diffusion network is a diffusion network in which contagions spread at the same rate α across different edges. That means, $\mathbf{A} \doteq [\alpha]$.

Weighted diffusion network. A weighted diffusion network is a diffusion network in which contagions spread at possibly different rates $\alpha_{i,j}$ across different edges. In other words, they allow for temporally heterogeneous interactions within the network, $\mathbf{A} \doteq [\alpha_{i,j}]$.

Pairwise transmission likelihood. The pairwise transmission likelihood $f(t_j|t_i;\alpha_{i,j})$ of an edge $(i,j) \in \mathcal{E}$ is the conditional likelihood of transmission from node i, activated at time t_i, to node j. The transmission likelihood depends on the activation times (t_i, t_j) and the transmission rate $\alpha_{i,j}$. A node cannot be activated by another node activated later in time. In other words, a node i that has been activated at a time t_i may activate a node j at a time t_j only if $t_i < t_j$, otherwise $f(t_j|t_i;\alpha_{i,j}) = 0$.

In the remainder of this dissertation, for simplicity, we consider several parametric time shift invariant transmission likelihoods, which result in parametric pairwise transmission models. We summarize them in Table 2.1.

In the power-law model, to have a bounded likelihood, we set δ as the minimum allowed time difference. Power-law and exponential are monotonic models that have been argued for in the literature (Barabási, 2005; Leskovec et al., 2007b; Malmgren et al., 2008). Power-laws model activations with long-tails. The Rayleigh and Weibull models are non-monotonic parametric models previously used in epidemiology (Kaplan, 1989; Wallinga and Teunis, 2004). They are well-adapted to modeling fads, where transmission likelihood rises to a peak and then drops extremely rapidly. In all four models, as $\alpha_{j,i} \to 0$, the likelihood of transmission tends to zero. Rigorously, our power-law model is a Pareto model. However, in the remainder of this dissertation, for clarity of exposition, we will name it power-law.

Pairwise survival function. The survival function $S(t_j|t_i;\alpha_{i,j})$ of an edge $(i,j) \in \mathcal{E}$ is the probability that node i, activated at time t_i, does *not* cause node j to activate by time t_j:

$$S(t_j|t_i;\alpha_{i,j}) = 1 - F(t_j|t_i;\alpha_{i,j}),$$

where $F(t_j|t_i;\alpha_{i,j})$ is the cumulative density function computed from the pairwise transmission likelihood. The pairwise survival function for the pairwise transmission models that we consider are simple, Table 2.1.

Hazard function. The hazard function, or instantaneous activation rate, of an edge

Model	**Transmission likelihood** $f(t_i \mid t_j; \alpha_{j,i})$		**Log survival** $\log S(t_i \mid t_j; \alpha_{j,i})$	**Hazard** $H(t_i \mid t_j; \alpha_{j,i})$
Exponential	$\alpha_{j,i} \cdot e^{-\alpha_{j,i}(t_i - t_j)}$	if $t_j < t_i$	$-\alpha_{j,i}(t_i - t_j)$	$\alpha_{j,i}$
	0	otherwise		
Power-law	$\frac{\alpha_{j,i}}{\delta}\left(\frac{t_i - t_j}{\delta}\right)^{-1-\alpha_{j,i}}$	if $t_j + \delta < t_i$	$-\alpha_{j,i} \log\left(\frac{t_i - t_j}{\delta}\right)$	$\alpha_{j,i} \frac{1}{t_i - t_j}$
	0	otherwise		
Rayleigh	$\alpha_{j,i}(t_i - t_j) e^{-\frac{1}{2}\alpha_{j,i}(t_i - t_j)^2}$	if $t_j < t_i$	$-\alpha_{j,i} \frac{(t_i - t_j)^2}{2}$	$\alpha_{j,i}(t_i - t_j)$
	0	otherwise		
Weibull	$k\alpha_{j,i}(t_i - t_j)^{k-1} e^{-\alpha_{j,i}(t_i - t_j)^k}$	if $t_j < t_i$	$-\alpha_{j,i}(t_i - t_j)^k$	$k\alpha_{j,i}(t_i - t_j)^{k-1}$
	0	otherwise		

Table 2.1: Pairwise transmission models

$(i, j) \in \mathcal{E}$ is the ratio

$$H(t_j \mid t_i; \alpha_{i,j}) = -\frac{S'(t_j \mid t_i; \alpha_{i,j})}{S(t_j \mid t_i; \alpha_{i,j})} = \frac{f(t_j \mid t_i; \alpha_{i,j})}{S(t_j \mid t_i; \alpha_{i,j})}.$$

The pairwise hazard function for the pairwise transmission models that we consider are simple, Table 2.1.

Prior transmission probability. The prior transmission probability $\beta_{i,j}$ of an edge (i, j) quantifies the probability that a contagion would eventually spread from node i to node j for arbitrarily large t_j. The prior transmission probability and the pairwise survival function are related through the equation $\beta_{i,j} = \lim_{t_j \to T} 1 - S(T \mid t_i; \alpha_{i,j})$, where T is large.

Influence function. Given a source set $\mathcal{S}$ and a time horizon T, the influence function $\sigma(\mathcal{S}; T)$ is defined as the average total number of nodes in $\mathcal{G}$ infected up to time T, *i.e.*, $\sigma(\mathcal{S}; T) = \mathbb{E}N(\mathcal{S}; T)$.

2.4 Sets, functions and distributions

In this section, we introduce several types of sets, functions and distributions that we will use throughout this dissertation.

Convex set. A set $C \subseteq \mathbb{R}^n$ is convex if, for any $x_1, x_2 \in C$ and any $t \in [0, 1]$, the point $(1-t)x_1 + tx_2 \in C$.

Convex and concave function. A real valued function $f : \mathbb{R}^n \to \mathbb{R}$ is called convex if its domain is a convex set and for any $x_1, x_2 \in dom f$ and any $t \in [0, 1]$,

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).$$

A function f is concave if $-f$ is convex.

Log-convex and log-concave function. A non-negative function $f : \mathbb{R}^n \to \mathbb{R}_+$ is log-convex if its domain is a convex set and for any $x_1, x_2 \in dom f$ and any $\theta \in [0, 1]$,

$$\log f(\theta x_1 + (1-\theta)x_2) \leq \theta \log f(x_1) + (1-\theta) \log f(x_2).$$

Similarly, a non-negative function f is log-concave if its domain is a convex set and for any $x_1, x_2 \in dom f$ and any $\theta \in [0, 1]$,

$$\log f(\theta x_1 + (1-\theta)x_2) \geq \theta \log f(x_1) + (1-\theta) \log f(x_2).$$

Finite set. A set $\mathcal{W}$ is called finite if there exists a bijection $f : \mathcal{W} \to \{1, \ldots, n\}$ for some natural number n. The number n is called the cardinality of the set, and is denoted $|\mathcal{W}|$.

Submodular function. A set function $f : 2^{\mathcal{W}} \to \mathbb{R}$ mapping subsets of a finite set $\mathcal{W}$ to the real numbers is submodular if whenever $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{W}$ and $s \in \mathcal{W} \setminus \mathcal{B}$, it holds that

$$f(\mathcal{A} \cup \{s\}) - f(\mathcal{A}) \geq f(\mathcal{B} \cup \{s\}) - f(\mathcal{B}),$$

or, in words, adding s to the set A provides a bigger marginal gain than adding s to the set B.

Heavy-tailed distributions. Heavy-tailed distributions are probability distributions whose tails are not exponentially bounded (Asmussen, 2003), *i.e.*, they have heavier tails than the exponential distribution. Mathematically, a random variable

X with cumulative distribution function $F(X)$ is said to have a heavy right tail if:

$$\lim_{x\to\infty} e^{\lambda x}(1 - F(X)) = \infty, \forall \lambda > 0.$$

Heavy-tailed distributions coincides with social, scientific, geophysical, actuarial, and many other types of observable phenomena (Ijiri and Simon, 1975; Seal, 1980; Faloutsos et al., 1999; Reed and Jorgensen, 2004; Schroeder et al., 2010). Throughout this dissertation, we use two types of heavy-tailed distributions, power-law and Weibull, to model pairwise interactions between nodes in a diffusion network, Table 2.1.

Phase-type distribution. A phase-type distribution is a probability distribution that results from a system of one or more inter-related Poisson processes occurring in sequence. Given any phase-type distribution, we can always find at least one (or more) Markov process(es) with one absorbing state in which the time until absorption follows such distribution. The cumulative distribution $F(x)$ and density function $f(x)$ of a phase-type distribution can be expressed as:

$$f(x) = [1\mathbf{0}]' e^{Bx} S^0 \text{ and } F(x) = 1 - [1\mathbf{0}]' e^{Bx} \mathbf{1},$$

where $Q = [B\, B^0; \mathbf{0}'\, 0]$ is the generator matrix of a Markov process, $B^0 = -B\mathbf{1}$, where $\mathbf{1}$ denotes an all 1's column vector and $\mathbf{0}$ an all 0's column vector and the matrix B results from removing the column and row associated to the last state of the underlying Markov process from Q. We can compute the mean as:

$$E(X) = -[1\mathbf{0}]' S^{-1} \mathbf{1}.$$

2.5 Real data

2.5.1 Memes and hyperlinks

We trace information spread on the Web using two types of contagions: *memes* and hyperlinks. A meme is a short unit text, a short distinctive phrase, a unit of information, or more generally an idea. Here, the collection of distinctive phrases that will act as tracers for memes are the set of quoted phrases and sentences that we find in articles, often attributed to individuals. Memes act as signatures of topics and events which propagate and diffuse over the web. We refer the reader to Leskovec et al. (2009) for a more extensive overview on memes on the context of information diffusion over the Web. Hyperlinks are used by blog or news sites to refer to other posts published by other sites. Therefore, we can consider them as events of information transmission. For each type of contagion, we develop a different methodology:

1. Hyperlink-based cascades We use hyperlinks between blog posts to trace the flow of information (Leskovec et al., 2007b). When a blog publishes a piece of information and uses hyper-links to refer to other posts published by other blogs we consider this as events of information transmission. A cascade $\mathbf{t}^c$ starts when a blog publishes a post P and the information propagates recursively to other blogs by them linking to the original post or one of the other posts. By following the chains of hyperlinks in the reverse direction we identify hyperlink cascades (Leskovec et al., 2007b). A cascade is thus composed of the time-stamps of the hyperlink/post creation times.

2. Meme-based cascades We use the MemeTracker (Leskovec et al., 2009) methodology to extract short textual phrases or memes (like, “Joe, the plumber” or “lipstick on a pig”). We then consider each textual phrases as a separate cascade $\mathbf{t}^c$. Since all documents are time stamped, a cascade $\mathbf{t}^c$ is simply a set of time-stamps when blogs first mentioned phrase c. So, we observe the times when blogs mention particular phrases but not where they copied or obtained the phrases from. Note that cascades in general do not spread over all the sites, which our methodology can successfully handle.

Figure 2.1 further illustrates the concept of hyperlink and meme cascades.

2.5.2 MemeTracker dataset (2008/09/01 $\rightarrow$ 2009/08/31)

The original MemeTracker dataset contains more than 172 million news articles and blog posts from 1 million online sources over a period of one year from September 1

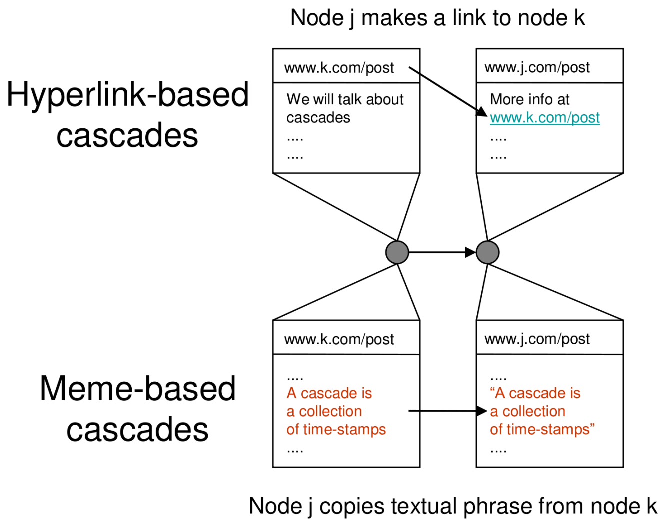

Node j makes a link to node k
Hyperlink-based cascades
www.k.com/post
We will talk about cascades
www.j.com/post
More info at www.k.com/post
Meme-based cascades
www.k.com/post
A cascade is a collection of time-stamps
www.j.com/post
"A cascade is a collection of time-stamps"
Node j copies textual phrase from node k

Figure 2.1: Hyperlink-based cascades versus meme-based cascades. In hyper-link cascades, if post j linked to post k, we consider this as a contagion transmission event with the post creation time as the corresponding infection time. In meme cascades, we follow the spread of a short textual phrase and use post creation times as infection times.

2008 till August 31 2009[1].

In this dataset, we trace the flow of information using memes and hyperlinks. We use the MemeTracker (Leskovec et al., 2009) methodology to extract more than 343 million short textual phrases or memes. Out of these, 8 million distinct phrases appeared more than 10 times, with the cumulative number of mentions of over 150 million. We refer the reader to Leskovec et al. (2009) for extensive overview and statistics on the original MemeTracker dataset.

2.5.3 Topic-based dataset (2011/03/01 → 2012/02/28)

The topic-based MemeTracker dataset contains more than 300 million news articles and blog posts from 3.3 million online sources over a period of one year, from March 2011 till February 2012[2].

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

[2]Data available at the `http://snap.stanford.edu/infopath/`

Table 2.2: Topic and news world event statistics

Topic or news event	**# sites**	**# recorded cascades**
Amy Winehouse	1,207	109,650
Fukushima	1,666	383,745
Gaddafi	1,358	440,646
Kate Middleton	1,427	191,777
NBA	2,087	1,543,630
Occupy	1,875	655,183
Strauss-Kahn	1,263	204,238
Syria	1,565	615,176

In this dataset, we also trace the flow of information using memes and hyperlinks. We used the MemeTracker (Leskovec et al., 2009) methodology to extract more than 179 million *memes*, longer than four words. Out of these, 34 million distinct *memes* appeared at least twice.

In contrast with the original MemeTracker dataset, described in the previous section, we assume we are also given a keyword query Q related to the event/topic of interest. When inferring a network for a given query Q, we only consider documents (and the memes they mention) that include keywords Q. Then, we build information cascades using only those memes and apply our algorithm to infer the edges and evolving edge transmission rates. The edge transmission rates explain the propagation of information related to a given topic or real world event Q. For each query Q we infer one network per day. Table 2.2 lists the number of sites and meme cascades for several topics and real world events that we used in Section 5.3.5. A comprehensive complete list for 38 different topics and real world events can be found at at the supporting website (InfoPath, 2013).

2.6 Table of symbols

Table 2.3 lists symbols which denote general network- and diffusion-theoretic concepts throughout this dissertation. Additionally, in every chapter, a chapter-specific table of symbols is included. A comprehensive complete list of symbols is included in the Appendix A.

Symbol	**Description**
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Directed network with node set $\mathcal{V}$ and edge set $\mathcal{E}$
$\mathbf{A}$	Pairwise transmission rates for all pair of nodes (i, j)
$(\mathcal{G}, \mathbf{A})$	Diffusion network: directed network $\mathcal{G}$ and transmission rates $\mathbf{A}$
$\alpha_{i,j}$	Pairwise transmission rate of edge (i, j)
c	Contagion
$\mathbf{t}^c$	Cascade: activation times for contagion c
$\mathcal{C}$	Set of all recorded cascades
t_i^c	Activation time of node i in cascade $\mathbf{t}^c$
T^c	Observation window cut-off or time horizon for cascade $\mathbf{t}^c$
$f(t_j \| t_i, \alpha_{i,j})$	Pairwise transmission likelihood of edge (i, j)
$F(t_j \| t_i, \alpha_{i,j})$	Cumulative density function of edge (i, j)
$S(t_j \| t_i; \alpha_{i,j})$	*Survival function* of edge (i, j)
$H(t_j \| t_i; \alpha_{i,j})$	*Hazard function*, or instantaneous activation rate, of edge (i, j)

Table 2.3: Table of symbols for general network- and diffusion-theoretic concepts.

Chapter 3

Survey of related work

3.1 Inference of diffusion networks

In recent years, information diffusion in on-line settings has received considerable attention (Gruhl et al., 2004; Kumar et al., 2004; Adar and Adamic, 2005; Leskovec et al., 2006a,b, 2007b; Liben-Nowell and Kleinberg, 2008), only a few studies were able to study the actual shapes of cascades (Leskovec et al., 2007b; Liben-Nowell and Kleinberg, 2008; Ghosh and Lerman, 2011; Romero et al., 2011b; Ver Steeg et al., 2011). The problem of inferring links of diffusion was first studied by Adar and Adamic (Adar and Adamic, 2005), who formulated it as a supervised classification problem and used Support Vector Machines combined with rich textual features to predict the occurrence of individual links. Although rich textual features are used, links are predicted independently and thus their approach is similar to the baseline method we introduced in Section 4.3.2 in the sense that it picks a threshold (*i.e.*, hyperplane in case of SVMs) and predicts individually the most probable links.

More recently, several diffusion network inference algorithms from cascade data, more closely related to our work, have been proposed (Saito et al., 2009; Myers and Leskovec, 2010; Saito et al., 2011; Snowsill et al., 2011; Netrapalli and Sanghavi, 2012; Wang et al., 2012; Du et al., 2012; Fyson et al., 2012). Some approaches consider unweighted networks, and infer only the network structure (Snowsill et al., 2011), while others consider weighted networks, and they do not only infer the network structure but also the *strength* or the average latency of every edge in the network (Saito et al., 2009; Myers and Leskovec, 2010; Wang et al., 2012; Du et al., 2012). Some of the approaches use submodular optimization (Gomez-Rodriguez et al., 2010), while others use convex optimization (Myers and Leskovec, 2010; Wang et al., 2012; Du et al., 2012) or expectation maximization (Saito et al., 2009, 2011). Most of the approaches use only temporal information while a few methods (Netrapalli and Sanghavi, 2012; Wang et al., 2012) consider both temporal information and additional non temporal features. Moreover, there have been also attempts to model information diffusion without assuming the existence of an underlying network (Yang and Leskovec, 2010, 2011; Bießmann et al., 2012).

Network structure learning has been also considered for estimating the dependency structure of probabilistic graphical models (Friedman and Koller, 2003; Friedman et al., 1999). However, there are fundamental differences between diffusion network inference and graphical models structure learning. First, diffusion network inference methods typically make no assumptions about the network structure (they allow cycles, reciprocal edges) and are thus able to learn general directed networks. In directed graphical models, reciprocal edges and cycles are not allowed, and the inferred network is a directed acyclic graph (DAG). In undirected graphical models, there are typically no assumptions about the network structure, but the inferred network is undirected. Second, Bayesian network structure inference methods are generally heuristic approaches without any approximation guarantees. Network structure learning has also been used for estimating epidemiological networks (Wallinga and Teunis, 2004) and for estimating probabilistic relational models (Getoor et al., 2003). In both cases, the problem is formulated in a probabilistic framework. However, since the problem is intractable, heuristic greedy hill-climbing or stochastic search that offer no performance guarantee were usually used in practice. In contrast, some diffusion network inference algorithms, including our work in Chapters 4 and 5, provide *tractable* solutions together with theoretical guarantees.

Diffusion network inference relates to static sparse graph estimation using graphical Lasso methods (Wainwright et al., 2006; Schmidt et al., 2007; Friedman et al., 2008; Meinshausen and Buehlmann, 2006), unsupervised structure network inference using kernel methods (Lippert et al., 2009), mutual information relevance network inference (Butte and Kohane, 2000), inference of influence probabilities (Goyal et al., 2010a), and extensions to time evolving graphical models (Ahmed and Xing, 2009; Ghahramani, 1998; Song et al., 2009). It is also related to a link prediction problem (Jansen et al., 2003; Taskar et al., 2003; Liben-Nowell and Kleinberg, 2003; Backstrom and Leskovec, 2011a; Vert and Yamanishi, 2005) but different in a sense that this line of work assumes that part of the network is already visible to us.

Last, although *submodular* function maximization has been previously considered for sensor placement (Leskovec et al., 2007a) and finding influencers in viral marketing (Kempe et al., 2003), to the best of our knowledge, NetInf and Multitree, which we describe in Chapter 4, are among the first that consider submodular function maximization in the context of network structure learning.

3.2 Influence maximization in diffusion networks

Richardson and Domingos (2002) were the first to study influence maximization as an algorithmic problem, motivated by marketing applications. In their work, they proposed heuristics for choosing a set of influential customers with a large overall effect on a network, and methods to infer the influence of each customer were developed.

Kempe, Kleinberg and Tardos (Kempe et al., 2003) posed influence maximization in a social network as a discrete optimization problem. They showed that the optimal solution is NP-hard for several models of influence, and obtained the first provable approximation guarantees for efficient algorithms based on a natural diminishing property of the problem, submodularity.

Since then there have been substantial developments that build on their seminal work. Efficient influence maximization that uses heuristics to speed up the optimization problem has been proposed (Chen et al., 2010, 2009) and influence maximization on the context of competing cascades (Bharathi et al., 2007; Budak et al., 2011; Carnes et al., 2007) or when negative opinions emerge (Chen et al., 2011) has been studied. Influence maximization has also been considered under additional constraints (Goyal et al., 2010b). It has been shown that minimizing the amount of available *time* (iterations) for the spread of influence or setting a *coverage* threshold on the number of influenced nodes also leads to discrete optimization problems with NP-hard optimal solutions. Recently, influence maximization has been studied in the context of game theory by introducing *influence games* (Irfan and Ortiz, 2011).

However, to the best of our knowledge, previous work on influence maximization has ignored the underlying temporal dynamics governing diffusion networks – once a transmission occurs, it always occurs at the same rate or temporal scale. In contrast, in our work (Chapter 6), we consider heterogeneous pairwise transmission rates since this is a key point in many real-world examples. In information propagation, news media sites and professional bloggers typically report news faster than people that maintain personal blogs. In epidemiology, people meet each other with different frequencies and then the pairwise transmission rates between individuals within a population differ. Finally, in viral marketing, some customers make up their minds about a product or service quicker than others, and then pass recommendations on to other customers at different rates.

Chapter 4

Inference of unweighted diffusion networks

4.1 Introduction

This chapter presents NetInf and Multitree, two methods for inferring static unweighted diffusion networks based on observed activations (Gomez-Rodriguez et al., 2010, 2012; Gomez-Rodriguez and Schölkopf, 2012c). To do so, we construct a probabilistic model incorporating some basic assumptions about the temporal structures that generate diffusion processes over unweighted networks. The assumptions are as follows:

A. activations are binary, *i.e.*, a node is either activated or it is not; we do not model partial activations or the partial propagation of information;

B. activations along edges of the network occur independently of each other;

C. information propagates through the network due only to diffusion, while ignoring any external sources;

D. cascades propagates independently of each other;

E. a node gets activated only by action of one parent; cascades of activations map to directed trees;

F. activation along edges of the network occur at the same activation rate; and,

G. we observe *all* activations occurring in the network during an arbitrarily large observation time window.

Assumptions (A-D) are common to our approach to inferring weighted diffusion networks in Chapter 5. However, assumptions (E-G) differ. Our aim is to infer the connectivity of the network after observing the times at which nodes in the network become activated.

In more detail, we formulate a generative probabilistic model in which, on a fixed hypothetical network, contagions spread as directed trees through the network. The model considers the information which propagates through the network due only to diffusion, while ignoring any external sources (Myers et al., 2012). 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. In order to infer the network we have to consider all possible ways of the contagion spreading through the network. Unfortunately, naive computation of the model takes exponential time since there is a combinatorially large number of propagation trees. We show that, perhaps surprisingly, computations over this super-exponential set of trees can be performed in quadratic time. We then propose two methods based on different approximations: NetInf and Multitree. Both methods rely on submodularity to find a near-optimal network with provable guarantees that best explain the observed cascades. Lazy evaluation (Leskovec et al., 2007a) and the local structure of the problem can be used to speed-up our method.

Our results on synthetic datasets show that we can reliably infer an unweighted network, regardless of the overall network structure. Validation on synthetic datasets shows that NetInf and Multitree outperforms a baseline heuristic by an order of magnitude and correctly discovers more than 90% of the edges. Moreover, Multitree outperforms NetInf when the number of observed cascades is small compared to the network size. We apply our algorithms to a real Web information propagation dataset of 170 million blog and news articles over a one year period. Our results show that online news propagation networks tend to have a core-periphery structure with

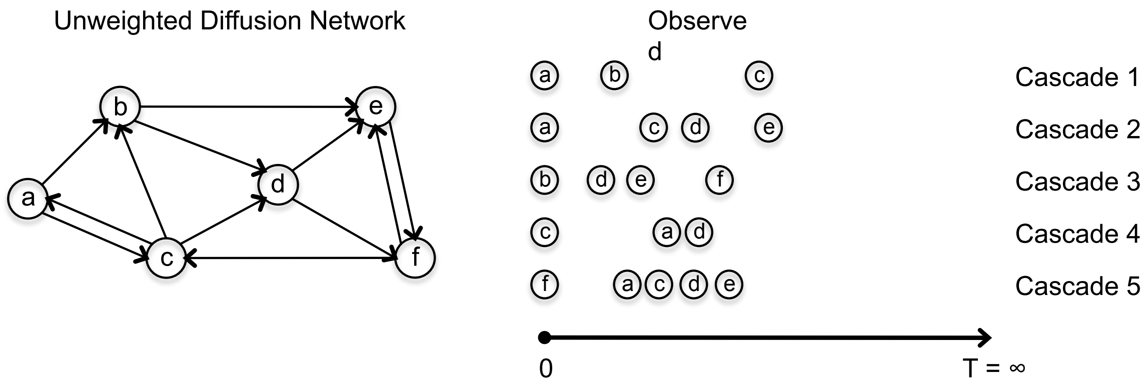


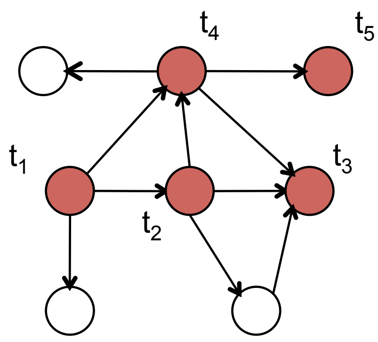
Figure 4.1: We observe a set of cascades (right) within an unknown unweighted diffusion network (left). For each contagion c, we observe the times in which nodes get infected but not who infected whom. Our goal is to infer the network $\mathcal{G}$ based on the observed cascades.

a small set of core blog and news media websites that diffuse information to the rest of the Web, news media websites tend to diffuse the news faster than blogs and blogs keep discussing about news longer time than media websites.

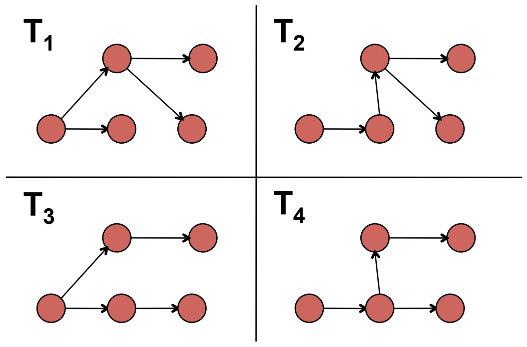
The remainder of the chapter is organized as follows: in Section 4.2, we describe our continuous time model of diffusion over unweighted networks and state the unweighted network inference problem. In Sections 4.3 and 4.4, we present two efficient inference algorithms for unweighted static network, NetInf and Multitree, and evaluate them on synthetic and real diffusion data. We conclude with a summary of our results in Section 4.5.

4.2 Problem formulation

In this section, we first describe the diffusion data our inference algorithms for unweighted diffusion networks are designed for and continue describing the generative model of diffusion. We conclude with a statement of the unweighted network inference problem.



(a) Cascade $\mathbf{t}^c$ on G



(b) Spanning trees induced by cascade $\mathbf{t}^c$ on $\mathcal{G}$

Figure 4.2: Panel (a) shows a cascade $\mathbf{t}^c = \{t_1, \ldots, t_5\}$ on network $\mathcal{G}$, where $t_{i-1} < t_i$. Panel (b) shows all connected spanning trees induced by cascade $\mathbf{t}^c$ on $\mathcal{G}$, *i.e.*, all possible ways in which a diffusion process spreading over $\mathcal{G}$ can create the cascade.

4.2.1 Data

We observe multiple waves of contagions that propagate on a fixed population of N nodes. As the contagion spreads from activated to non-activated nodes it creates a *cascade*. For each contagion c, we observe a cascade $\mathbf{t}^c$, which is simply a record of observed node activation times. In an information propagation setting, each cascade corresponds to a different piece of information and the activation time of a node is simply the time when the node first heard of or mentioned the piece of information.

We record a set $\mathcal{C}$ of cascades $\{\mathbf{t}^1, \ldots, \mathbf{t}^{|\mathcal{C}|}\}$. A cascade $\mathbf{t}^c = (t_1^c, \ldots, t_N^c)$ is an N-dimensional vector recording when each of N nodes got activated by the contagion c during a time interval of length arbitrarily large, $T^c \to \infty$. Thus, $t_k^c \in [t_0, \infty) \cup \{\infty\}$, where t_0 is the activation time of the first node. Symbol ∞ labels nodes that are not activated by the contagion c. We assume contagions spread at the same rate α over different edges of the underlying unobserved network $\mathcal{G}$. Thus, we consider the network to be unweighted. Contagions often propagate simultaneously (Myers and Leskovec, 2012; Prakash et al., 2012b) over the same network but we assume each contagion to propagate independently of each other. We illustrate this process in Figure 4.1.

Given a set of node activation times of many different contagions, our goal is to

Symbol	Description
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Directed network with node set $\mathcal{V}$ and edge set $\mathcal{E}$
$\mathbf{A} = [\alpha]$	Pairwise transmission rates for all pair of nodes (i, j)
$(\mathcal{G}, \mathbf{A})$	Diffusion network: directed network $\mathcal{G}$ and transmission rates $\mathbf{A}$
$\alpha_{i,j} = \alpha$	Pairwise transmission rate of edge (i, j)
$\beta_{i,j} = \beta$	Prior transmission probability of edge (i, j)
$f(t_j \| t_i, \alpha)$	Pairwise transmission likelihood of edge (i, j)
c	Contagion
$\mathbf{t}^c$	Cascade: activation times for contagion c
$\mathcal{C}$	Set of all recorded cascades
t_i^c	Activation time of node i in cascade $\mathbf{t}^c$
$T^c = \infty$	Observation window cut-off or time horizon for cascade $\mathbf{t}^c$
$\mathcal{E}_\varepsilon$	Set of ε-edges, $\mathcal{E} \cap \mathcal{E}_\varepsilon = \emptyset$ and $\mathcal{E} \cup \mathcal{E}_\varepsilon = \mathcal{V} \times \mathcal{V}$
$\mathcal{T}_c(\mathcal{G})$	Set of all possible propagation trees of cascade $\mathbf{t}^c$ on network $\mathcal{G}$
$\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$	Cascade propagation tree, $\mathcal{T} \in \mathcal{T}_c(\mathcal{G})$
$\mathcal{V}_\mathcal{T}$	Node set of T, $\mathcal{V}_\mathcal{T} = \{i \mid i \in \mathcal{V}\text{and } \mathbf{t}_c[i] < \infty\}$
$\mathcal{E}_\mathcal{T}$	Edge set of T, $\mathcal{E}_\mathcal{T} \subseteq \mathcal{E} \cup \mathcal{E}_\varepsilon$

Table 4.1: Table of symbols for Chapter 4.

discover the unknown, unobserved network $\mathcal{G}$ over which cascades originally propagated. Importantly, the time-stamps assigned to nodes in each cascade induce the structure of a directed acyclic graph (DAG) on the network (which is *not* acyclic in general). Thus, it is meaningful to refer to parents and children within a cascade, but not on the network. The DAG structure dramatically simplifies the computational complexity of the inference problem.

4.2.2 Pairwise interactions

We describe the pairwise interactions between nodes using three concepts: pairwise transmission rates $\alpha_{i,j}$, prior probabilities of transmission $\beta_{i,j}$, and pairwise transmission likelihoods $f(t_i|t_j, \alpha_{i,j})$. The transmission rate $\alpha_{i,j}$ of an edge $(i, j) \in \mathcal{E}$ quantifies how frequently any contagion spreads from node i to node j or, in other words, the *latency* of the edge (i, j). The prior transmission probability $\beta_{i,j}$ of an edge (i, j) quantifies the probability that a contagion would eventually spread from node i to node j for arbitrarily large t_j. Finally, the pairwise transmission likelihood

$f(t_j|t_i;\alpha_{i,j})$ of an edge $(i,j) \in \mathcal{E}$ is the conditional likelihood of transmission from node i, activated at time t_i, to node j. We refer the reader to Section 2.3 for an in-depth discussion of all three concepts. Now, we highlight the key assumptions on the pairwise interactions of our model of diffusion over unweighted networks:

First, since we assume networks to be unweighted, we consider activations to occur at the same rate over different edges of a network. That means, $\alpha_{j,i} = \alpha$ for every edge $(i,j) \in \mathcal{E}$.

Second, for any contagion c, we observe activations up to an arbitrarily large time horizon $T^c \to \infty$. Then, we need to assume the prior probability of transmission $\beta_{i,j}$ to be typically smaller than 1. Otherwise, given a node i, activated at t_i, and an edge (i,j), we would always observe node j to get infected at some (arbitrarily large) time t_j. This contrasts with our approach to diffusion over weighted networks in Section 5, where we consider finite time horizons $T^c < \infty$.

Third, the transmission likelihood for every edge (i,j) depends on the activation times (t_i,t_j) and a unique transmission rate α. The shape of the conditional likelihood of transmission may depend on the particular setting (information, influence, diseases, etc.) in which propagation takes place. In some scenarios, it may be possible to estimate a non-parametric likelihood while in others, expert knowledge may be used to decide upon a parametric model. For simplicity, we consider well-known parametric models used previously in the literature (refer to Table 2.1). However, our approach to inference of unweighted networks allows for the transmission likelihood to be arbitrarily complicated and only requires $f(t_j|t_i;\cdot) = 0$ for $t_i > t_j$.

4.2.3 Likelihood of a cascade for a given tree

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

$$f(\mathbf{t}^c|\mathcal{T}) = \prod_{(i,j)\in\mathcal{E}_T} f(t_j|t_i;\alpha), \tag{4.1}$$

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

4.2.4 Probability of a tree in a given network

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

$$P(\mathcal{T}|\mathcal{G}) = \prod_{(i,j)\in\mathcal{E}_T} \beta \prod_{u\in\mathcal{V}_T,(u,x)\in\mathcal{E}\setminus\mathcal{E}_T} (1-\beta) = \beta^q(1-\beta)^r, \tag{4.2}$$

where $\mathcal{V}_T$ is the vertex set of tree $\mathcal{T}$, $\mathcal{E}_T$ is the edge set of tree $\mathcal{T}$, $\mathcal{E}$ is the edge set of the network $\mathcal{G}$, $q = |\mathcal{E}_T| = |\mathcal{V}_T| - 1$ is the number of edges in $\mathcal{T}$ and counts the edges over which the diffusion process successfully propagated, and r counts the number of edges that did not activate and failed to transmit the contagion: $r = \sum_{u\in\mathcal{V}_T} d_{out}(u) - q$, and $d_{out}(u)$ is the out-degree of node u in graph $\mathcal{G}$. For a particular cascade $\mathbf{t}^c$ and tree $\mathcal{T}$, $\mathcal{V}_T$ is the set of nodes whose activation times $t_i < \infty$. The first product accounts for the *active* edges in $\mathcal{G}$, *i.e.*, edges that define the tree $\mathcal{T}$, and the second product accounts for the *inactive* edges in $\mathcal{G}$, *i.e.*, edges where the contagion did not spread.

4.2.5 Likelihood of a cascade in a given network

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

$$f(\mathbf{t}^c|\mathcal{G}) = \sum_{\mathcal{T}\in\mathcal{T}_c(\mathcal{G})} f(\mathbf{t}^c|\mathcal{T})P(\mathcal{T}|\mathcal{G}), \tag{4.3}$$

where $\mathcal{T}_c(\mathcal{G})$ is the set of all the directed connected spanning trees on the subnetwork of $\mathcal{G}$ induced by the nodes that got activated in cascade $\mathbf{t}^c$, *i.e.*, $t_i \in \mathbf{t}^c : t_i < \infty$. Figure 4.2 illustrates the notion of a cascade and all the connected spanning trees $\mathcal{T}$ induced by its nodes.

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

$$f(\mathbf{t}^1, \ldots, \mathbf{t}^{|\mathcal{C}|}|G) = \prod_{\mathbf{t}^c \in \mathcal{C}} f(\mathbf{t}^c|\mathcal{G}). \tag{4.4}$$

Importantly, the likelihood of a cascade $\mathbf{t}^c$ under our diffusion model for weighted networks, given by Eq. 5.8, relates to likelihood under our diffusion model for unweighted networks, given by Eq. 4.3, by considering arbitrarily large time horizons T^c and equal transmission rates $\alpha_{i,j} = \alpha$. We refer the reader to Section 5.2.5 for further details on the actual relationship.

4.2.6 The unweighted network inference problem

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

$$\hat{\mathcal{G}} = \underset{|\mathcal{G}| \leq k}{\operatorname{argmax}}\, f(\mathbf{t}^1, \ldots, \mathbf{t}^N|\mathcal{G}), \tag{4.5}$$

where the maximization is over all directed networks $\mathcal{G}$ of at most k edges. We include the constraint on the number of edges in $\hat{\mathcal{G}}$ simply because we seek for a sparse solution, since real networks are sparse. We discuss how to choose k in further sections of the chapter.

4.3 NetInf

4.3.1 Algorithm

At first, the optimization problem in Eq. 4.5 seems wildly intractable. We now propose an alternative formulation of the problem that is tractable both to compute and also to optimize. We use the same tree cascade formation model as in the previous section. However, we compute an approximation of the likelihood of a single cascade by considering only the most likely tree instead of all possible propagation trees. We show that this approximate likelihood is tractable to compute. Moreover, we also devise an algorithm that provably finds networks with near optimal approximate likelihood. In the remainder of this section, we informally write likelihood and log-likelihood even though they are approximations. However, all approximations are clearly indicated.

First we introduce the concept of ε-edges to account for the fact that nodes may get activated for reasons other than the network influence. For example, in online media, not all the information propagates via the network, as some is also pushed onto the network by the mass media (Katz and Lazarsfeld, 1955; Watts and Dodds, 2007) and thus a disconnected cascade can be created. Similarly, in viral marketing, a person may purchase a product due to the influence of peers (*i.e.*, network effect) or for some other reason (*e.g.*, seing a commercial on TV) (Leskovec et al., 2006a).

Modeling external influence via ε-edges

To account for such phenomena when a cascade "jumps" across the network we can think of creating an additional node x that represents an *external influence* and can activate *any* other node u with small probability. We then connect the external influence node x to every other node u with an ε-edge. And then every node u can get activated by the external source x with a very small probability ε. For example, in case of information diffusion in the blogosphere, such a node x could model the effect of blogs getting activated by the mainstream media.

However, if we were to adopt this approach and insert an additional external influence node x into our data, we would also need to infer the edges pointing out of x, which would make our problem even harder. Thus, in order to capture the effect of external influence, we introduce a concept of ε-edge. If there is not a network edge between a node i and a node j in the network, we add an ε-edge and then node i can activate node j with a small probability ε. Even though adding ε-edges makes our network $\mathcal{G}$ a clique (*i.e.*, the union of network edges and ε-edges creates a clique), the ε-edges play the role of external influence node. It is important to remark that adding ε-edges 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 assumed to be activated by an external source. We will study this tradeoff experimentally in Section 4.3.2.

Thus, we now think of network $\mathcal{G}$ as a fully connected network of two disjoint sets of edges, the network edge set $\mathcal{E}$ and the ε-edge set $\mathcal{E}_\varepsilon$, *i.e.*, $\mathcal{E} \cap \mathcal{E}_\varepsilon = \emptyset$ and $\mathcal{E} \cup \mathcal{E}_\varepsilon = \mathcal{V} \times \mathcal{V}$.

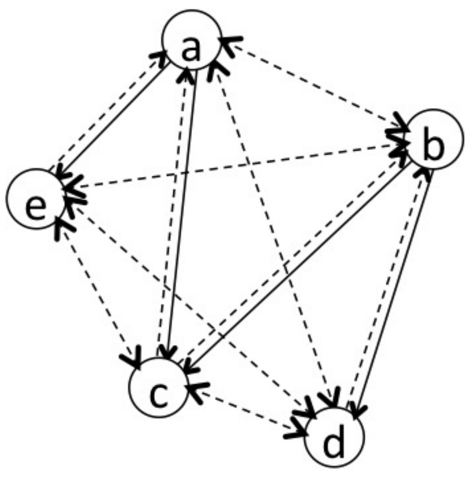
Now, any cascade propagation tree $\mathcal{T}$ is a combination of network and ε-edges. As we model the external influence via the ε-edges, the probability of a tree (*i.e.*, the analog of Eq. 4.2) can now be computed as products of edge-types:

$$P(\mathcal{T}|\mathcal{G}) = \beta^q \, \varepsilon^{q'} \, (1-\beta)^s \, (1-\varepsilon)^{s'} \approx \beta^q \, \varepsilon^{q'} \, (1-\varepsilon)^{s+s'}, \tag{4.6}$$

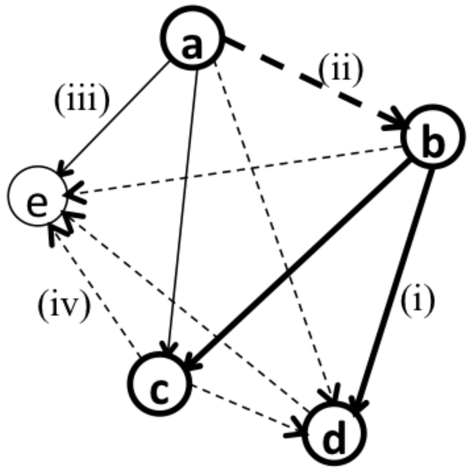
where q is the number of network edges in $\mathcal{T}$ (type (a) edges in Fig. 4.3(b)), q' is the number of ε-edges in $\mathcal{T}$, s is the number of network edges that did not transmit and s' is the number of ε-edges that did not transmit. Note that the above approximation is valid since real networks are sparse and cascades are generally small, and hence $s' \gg s$. Thus, even though $\beta \gg \varepsilon$ we expect $(1-\beta)^s$ to be of about same order of magnitude as $(1-\varepsilon)^{s'}$.

Note that above we distinguish four type of edges: network and ε-edges that participated in the diffusion of the contagion and network and ε-edges that did not participate in the diffusion.

Figure 4.3 further illustrates this concept. First, Figure 4.3(a) shows an example of a network $\mathcal{G}$ on five nodes and four network edges $\mathcal{E}$ (solid lines), and any other



(a) Network $\mathcal{G}$ on five vertices and four network edges (solid edges). ε-edges shown as dashed lines.



(b) Cascade propagation tree $\mathcal{T} = \{(a,b),(b,c),(b,d)\}$

Figure 4.3: (a) Network $\mathcal{G}$: Network edges $\mathcal{E}$ are shown as solid, and ε-edges are shown as dashed lines. (b) Propagation tree $\mathcal{T} = \{(a,b),(b,c),(b,d)\}$. Four types of edges are labeled: (i) network edges that transmitted the contagion (solid bold), (ii) ε-edges that transmitted the contagion (dashed bold), (iii) network edges that failed to transmit the contagion (solid), and (iv) ε-edges that failed to transmit the contagion (dashed).

possible edge is the ε-edge (dashed lines). Then, Figure 4.3(b) shows an example of a propagation tree $\mathcal{T} = \{(a,b),(b,c),(b,d)\}$ in network G. We only show the edges that play a role in Eq. 4.6 and label them with four different types: (a) network edges that transmitted the contagion, (b) ε-edges that transmitted the contagion, (c) network edges that failed to transmit the contagion, and (d) ε-edges that failed to transmit the contagion.

We can now rewrite the likelihood of a cascade in a given network $f(\mathbf{t}^c|\mathcal{G})$ by combining Eq. 4.1 and Eq. 4.6:

$$f(\mathbf{t}^c|\mathcal{G}) \approx \prod_{(i,j)\in E_{\mathcal{T}}} \beta^q \ \varepsilon^{q'} \ (1-\varepsilon)^{s+s'} f(t_j|t_i;\alpha) = \prod_{(i,j)\in E_{\mathcal{T}}} f'(t_j|t_i;\alpha,\beta,\varepsilon), \tag{4.7}$$

where we introduce $f'(t_j|t_i;\alpha,\beta,\varepsilon) = \beta^q \ \varepsilon^{q'} \ (1-\varepsilon)^{s+s'} f(t_j|t_i;\alpha)$ for mathematical convenience.

The formulation in Equation 4.7 has several benefits. Due to the introduction of ε-edges the likelihood $f(\mathbf{t}^c|\mathcal{T})$ is always positive. For example, even if we consider network G with no edges, $f(\mathbf{t}^c|\mathcal{T})$ is still well defined as we can explain the tree $\mathcal{T}$ via the diffusion over the ε-edges. A second benefit that will become very useful later is that the likelihood now becomes monotonic in the network edges of $\mathcal{G}$. This means that adding an edge to $\mathcal{G}$ (*i.e.*, converting ε-edge into a network edge) only increases the likelihood.

Considering only the most likely propagation tree

So far we introduced the concept of ε-edges to model the external influence or diffusion that is exogenous to the network, and introduce an approximation to treat all edges that did not participate in the diffusion as ε-edges.

Now we consider the last approximation, where instead of considering all possible cascade propagation trees $\mathcal{T}$, we only consider the most likely cascade propagation trees $\mathcal{T}$:

$$f(\mathbf{t}^1, \dots, \mathbf{t}^{|\mathcal{C}|}|G) \approx \prod_{c \in \mathcal{C}} \max_{T \in \mathcal{T}_c(\mathcal{G})} \prod_{(i,j) \in E_{\mathcal{T}}} f'(t_j|t_i; \alpha, \beta, \varepsilon) \tag{4.8}$$

Thus now we aim to solve the network inference problem by finding a network G that maximizes Equation 4.8.

This formulation simplifies the original network inference problem by considering the most likely (*best*) propagation tree $\mathcal{T}$ per cascade $\mathbf{t}^c$ instead of considering all possible propagation trees $\mathcal{T}$ for each cascade c. Although in some cases we expect the likelihood of $\mathbf{t}^c$ with respect to the true tree $\mathcal{T}'$ to be much higher than that with respect to any competing tree $\mathcal{T}''$ and thus the probability mass will be concentrated at $\mathcal{T}'$, there might be some cases in which the probability mass does not concentrate on one particular $\mathcal{T}$. However, we run extensive experiments on small networks with different structures in which both the original network inference problem and the alternative formulation can be solved using exhaustive search. Our experimental results looked really similar and the results were indistinguishable. Therefore, we consider our approximation to work well in practice.

For convenience, we work with the log-likelihood $\log f(\mathbf{t}^c|\mathcal{G})$ rather than likelihood $f(\mathbf{t}^c|\mathcal{G})$. Moreover, instead of directly maximizing the log-likelihood we equivalently maximize the following objective function that defines the improvement of log-likelihood for cascade $\mathbf{t}^c$ occurring in network $\mathcal{G}$ over $\mathbf{t}^c$ occurring in an empty network $\bar{\mathcal{K}}$ (*i.e.*, network with only ε-edges and no network edges):

$$F(\mathbf{t}^c|\mathcal{G}) = \max_{T\in\mathcal{T}_c(\mathcal{G})} \log f'(t_j|t_i;\alpha,\beta,\varepsilon) - \max_{T\in\mathcal{T}_c(\bar{\mathcal{K}})} \log f'(t_j|t_i;\alpha,\beta,\varepsilon). \tag{4.9}$$

Maximizing Eq. (4.8) is equivalent to maximizing the following log-likelihood function:

$$F(\mathcal{C}|\mathcal{G}) = F(\mathbf{t}^1,\ldots,\mathbf{t}^{|\mathcal{C}|}|\mathcal{G}) = \sum_{c\in\mathcal{C}} F(\mathbf{t}^c|\mathcal{G}). \tag{4.10}$$

We now expand Eq. (4.9) and obtain an instance of a *simplified diffusion network inference problem*:

$$\hat{\mathcal{G}} = \arg\max_{\mathcal{G}} F(\mathcal{C}|\mathcal{G}) = \sum_{c\in\mathcal{C}} \max_{T\in\mathcal{T}_c(\mathcal{G})} \sum_{(i,j)\in\mathcal{E}_T} \log w_c(i,j), \tag{4.11}$$

where $w_c(i,j) = \varepsilon^{-1} f'(t_j|t_i;\alpha,\beta,\varepsilon)$ is a non-negative weight which can be interpreted as the improvement in log-likelihood of edge (i,j) under the most likely propagation tree $\mathcal{T}$ in $\mathcal{G}$. Note that by the approximation in Equation 4.7 one can ignore the contribution of edges that did not participate in a particular cascade $\mathbf{t}^c$. The contribution of these edges is constant, *i.e.*, independent of the particular shape that propagation tree $\mathcal{T}$ takes. This is due to the fact that each spanning tree $\mathcal{T}$ of $\mathcal{G}$ with node set $\mathcal{V}_T$ has $|\mathcal{V}_T| - 1$ (network and ε-) edges that participated in the cascade, and all remaining edges stopped the cascade from spreading. The number of non-spreading edges depends only on the node set $\mathcal{V}_T$ but *not* the edge set $\mathcal{E}_T$. Thus, the tree $\mathcal{T}$ that maximizes $f(\mathbf{t}^c|\mathcal{G})$ also maximizes $\sum_{(i,j)\in E_{\mathcal{T}}} \log w_c(i,j)$.

Since $\mathcal{T}$ is a tree that maximizes the sum of the edge weights this means that the most likely propagation tree $\mathcal{T}$ is simply the *maximum weight directed spanning tree* of nodes $\mathcal{V}_T$, where each edge (i,j) has weight $\log w_c(i,j)$, and $F(\mathbf{t}^c|\mathcal{G})$ is simply the sum of the weights of edges in $\mathcal{T}$.

We also observe that since edges (i, j) where $t_i \geq t_j$ have weight 0 (*i.e.*, such edges are not present) then the outgoing edges of any node u only point forward in time, *i.e.*, a node can not activate already activated nodes. Thus for a fixed cascade c, the collection of edges with positive weight forms a directed *acyclic* network or directed acyclic graph (DAG).

Now we use the fact that the collection of edges with positive weights forms a directed acyclic graph by observing that the maximum weight directed spanning tree of a DAG can be computed efficiently:

Proposition 1. *In a DAG $D(\mathcal{V}, \mathcal{E}, w)$ with vertex set $\mathcal{V}$ and nonnegative edge weights w, the maximum weight directed spanning tree can be found by choosing, for each node v, an incoming edge (u, v) with maximum weight $w(u, v)$.*

Proof. The score

$$S(\mathcal{T}) = \sum_{(i,j) \in T} w(i, j) = \sum_{i \in \mathcal{V}} w(Par_T(i), i)$$

of a tree $\mathcal{T}$ is the sum of the incoming edge weights $w(Par_T(i), i)$ for each node i, where $Par_T(i)$ is the parent of node i in $\mathcal{T}$ (and the root is handled appropriately). Now,

$$\max_T S(\mathcal{T}) = \max_T \sum_{(i,j) \in T} w(i, j) = \sum_{i \in \mathcal{V}} \max_{Par_T(i)} w(Par_T(i), i).$$

Latter equality follows from the fact that since $\mathcal{G}$ is a DAG, the maximization can be done independently for each node without creating any cycles. □

This proposition is a special case of the more general maximum spanning tree (MST) problem in directed networks (Edmonds, 1967). The important fact now is that we can find the best propagation tree $\mathcal{T}$ in time $O(|\mathcal{V}_T| D_{in})$, i.e., linear in the number of edges and the maximum in-degree $D_{in} = \max_{u \in \mathcal{V}_T} d_{in}(u)$ by simply selecting an incoming edge of highest weight for each node $u \in \mathcal{V}_T$. Algorithm 1 provides the pseudocode to efficiently compute the maximum weight directed spanning tree of a DAG.

Algorithm 1 Maximum weight directed spanning tree of a DAG

```
Require: Weighted directed acyclic graph $D(\mathcal{V}, \mathcal{E}, w)$
  $\mathcal{T} \leftarrow \{\}$
  for all $i \in \mathcal{V}$ do
    $Par_T(i) = \arg\max_j w(j, i)$
    $\mathcal{T} \leftarrow T \cup \{(Par_T(i), j)\}$
  end for
  return $\mathcal{T}$
```

Putting it all together we have shown how to efficiently evaluate the log-likelihood $F(\mathcal{C}|\mathcal{G})$ of a network $\mathcal{G}$. To find the most likely tree $\mathcal{T}$ for a single cascade takes $O(|\mathcal{V}_T|D_{in})$, and this has to be done for a total of $|\mathcal{C}|$ cascades. Interestingly, this is independent of the size of network $\mathcal{G}$ and only depends on the amount of observed data (*i.e.*, size and the number of cascades).

Now we aim to find network $\mathcal{G}$ that maximizes the log-likelihood $F(\mathcal{C}|\mathcal{G})$. First we notice that by construction $F(\mathcal{C}|\bar{\mathcal{K}}) = 0$, *i.e.*, the empty network has score 0. Moreover, we observe that the objective function $F(\cdot|\mathcal{G})$ is non-negative and monotonic. This means that $F(\mathcal{C}|\mathcal{G}) \leq F(\mathcal{C}|\mathcal{G}')$ for networks $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and $\mathcal{G}'(\mathcal{V}, \mathcal{E}')$, where $\mathcal{E} \subseteq \mathcal{E}'$. Hence adding more edges to $\mathcal{G}$ does not decrease the solution quality, and thus the complete network maximizes $F(\cdot|\mathcal{G})$. Monotonicity can be shown by observing that, as edges are added to $\mathcal{G}$, ε-edges are converted to network edges, and therefore the weight of any tree (and therefore the value of the maximum spanning tree) can only increase. However, since real-world social and information networks are usually sparse, we are interested in inferring a *sparse* network $\mathcal{G}$, that only contains some small number k of edges. Thus we aim to solve:

Problem 1. *Given the activation times of a set of cascades $\mathcal{C}$, probability of propagation β and the pairwise transmission likelihood $f(t_j|t_i;\alpha)$, find $\hat{\mathcal{G}}$ that maximizes:*

$$\mathcal{G}^* = \underset{|\mathcal{G}| \leq k}{\operatorname{argmax}}\, F(\mathcal{C}|\mathcal{G}), \tag{4.12}$$

where the maximization is over all networks $\mathcal{G}$ of at most k edges, and $F(\mathcal{C}|\mathcal{G})$ is defined by Eqs. 4.10 and 4.11.

Naively searching over all k edge networks would take time exponential in k, which is intractable. Moreover, finding the optimal solution to Eq. (4.12) is NP-hard, so we cannot expect to find the optimal solution:

Theorem 1. *The network inference problem defined by equation* (4.12) *is NP-hard.*

Proof. By reduction from the MAX-k-COVER problem (Khuller et al., 1999). In MAX-k-COVER, we are given a finite set $\mathcal{W}$, $|\mathcal{W}| = n$ and a collection of subsets $\mathcal{S}_1, \ldots, \mathcal{S}_m \subseteq \mathcal{W}$. The function

$$F_{MC}(A) = | \cup_{i \in A} \mathcal{S}_i |$$

counts the number of elements of $\mathcal{W}$ covered by sets indexed by A. Our goal is to pick a collection of k subsets A maximizing F_{MC}. We will produce a collection of n cascades $\mathcal{C}$ over a network $\mathcal{G}$ such that $\max_{|\mathcal{G}| \leq k} F(\mathcal{C}|\mathcal{G}) = \max_{|A| \leq k} F_{MC}(A)$. Network $\mathcal{G}$ will be defined over the set of vertices $\mathcal{V} = \{1, \ldots, m\} \cup \{r\}$, *i.e.*, there is one vertex for each set S_i and one extra vertex r. For each element $s \in \mathcal{W}$ we define a cascade which has time stamp 0 associated with all nodes $i \in \mathcal{V}$ such that $s \in S_i$, time stamp 1 for node r and ∞ for all remaining nodes.

Furthermore, we can choose the transmission model such that $\log w_c(i, r) = 1$ whenever $s \in S_i$ and $\log w_c(i', j') = 0$ for all remaining edges (i', j'), by choosing the parameters ε, α and β appropriately. Since a directed spanning tree over a network $\mathcal{G}$ can contain at most one edge incoming to node r, its weight will be 1 if $\mathcal{G}$ contains any edge from a node i to r where $s \in S_i$, and 0 otherwise. Thus, a network $\mathcal{G}$ of at most k edges corresponds to a feasible solution $A_{\mathcal{G}}$ to MAX-k-COVER where we pick sets S_i whenever edge $(i, r) \in \mathcal{G}$, and each solution A to MAX-k-COVER corresponds to a feasible solution $\mathcal{G}_A$ of (4.12). Furthermore, by construction, $F_{MC}(A_{\mathcal{G}}) = F(\mathcal{C}|\mathcal{G})$. Thus, if we had an efficient algorithm for deciding whether there exists a network $\mathcal{G}$, $|\mathcal{G}| \leq k$ such that $F(\mathcal{C}|\mathcal{G}) > c$, we could use the algorithm to decide whether there exists a solution A to MAX-k-COVER with value at least c. □

While finding the optimal solution is hard, we now show that $F(\cdot|\mathcal{G})$ satisfies *submodularity*, a natural diminishing returns property. Refer to Section 2.4 for a definition of submodularity. Submodularity allows us to efficiently find a *provably near-optimal* solution to this otherwise NP-hard optimization problem:

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

Proof. Fix a cascade $\mathbf{t}^c$, networks $\mathcal{G} \subseteq \mathcal{G}'$ and an edge $e = (r, s)$ not contained in $\mathcal{G}'$. We will show that $F(\mathcal{C}|\mathcal{G} \cup \{e\}) - F(\mathcal{C}|\mathcal{G}) \geq F(\mathcal{C}|\mathcal{G}' \cup \{e\}) - F(\mathcal{C}|\mathcal{G}')$. Since nonnegative linear combinations of submodular functions are submodular, the function $F(\mathcal{C}| = \sum_{c \in C} F(\mathbf{t}^c|G)$ is submodular as well. Let $w_{i,j}$ be the weight of edge (i, j) in $\mathcal{G} \cup \{e\}$, and $w'_{i,j}$ be the weight in $\mathcal{G}' \cup \{e\}$. As argued before, the maximum weight directed spanning tree for DAGs is obtained by assigning to each node the incoming edge with maximum weight. Let (i, s) be the edge incoming at s of maximum weight in $\mathcal{G}$, and (i', s) the maximum weight incoming edge in $\mathcal{G}'$. Since $\mathcal{G} \subseteq \mathcal{G}'$ it holds that $w_{i,s} \leq w'_{i',s}$. Furthermore, $w_{r,s} = w'_{r,s}$. Hence,

$$\begin{aligned} F(\mathbf{t}^c|\mathcal{G} \cup \{(r,s)\}) - F(\mathbf{t}^c|\mathcal{G}) &= \max(w_{i,s}, w_{r,s}) - w_{i,s} \\ &\geq \max(w'_{i',s}, w'_{r,s}) - w'_{i',s} \\ &= F(\mathbf{t}^c|\mathcal{G}' \cup \{(r,s)\}) - F(\mathbf{t}^c|\mathcal{G}'), \end{aligned}$$

proving submodularity of $F(\mathbf{t}^c|\mathcal{G})$ on $\mathcal{G}$. □

Maximizing submodular functions in general is NP-hard (Khuller et al., 1999). A commonly used heuristic is the *greedy algorithm*, which starts with an empty network $\bar{\mathcal{K}}$, and iteratively, in step i, adds the edge e_i which maximizes the marginal gain:

$$e_i = \underset{e \in \mathcal{G} \setminus \mathcal{G}_{i-1}}{\operatorname{argmax}} F(\mathcal{C}|G_{i-1} \cup \{e\}) - F(\mathcal{C}|G_{i-1}). \tag{4.13}$$

The algorithm stops once it has selected k edges, and returns the solution $\hat{\mathcal{G}} = \{e_1, \ldots, e_k\}$. The stopping criteria, *i.e.*, value of k, can be based on some threshold of the marginal gain, of the number of estimated edges or another more sophisticated heuristic.

In our context we can think about the greedy algorithm as starting on an empty network $\bar{\mathcal{K}}$ with no network edges. In each iteration i, the algorithm adds to $\mathcal{G}$ the edge e_i that currently improves the most the value of the log-likelihood. Another way to view the greedy algorithm is that it starts on a fully connected network $\bar{\mathcal{K}}$ where all the edges are ε-edges. Then adding an edge to network $\mathcal{G}$ corresponds to that edge changing the type from ε-edge to a network edge. Thus our algorithm iteratively swaps ε-edges to network edges until k network edges have been swapped (*i.e.*, inserted into the network $\mathcal{G}$).

Guarantees on the solution quality

Considering the NP-hardness of the problem, we might expect the greedy algorithm to perform arbitrarily bad. However, we will see that this is not the case. A fundamental result of Nemhauser et al. (1978) proves that for monotonic submodular functions, the set $\hat{\mathcal{G}}$ returned by the greedy algorithm obtains at least a constant fraction of $(1 - 1/e) \approx 63\%$ of the optimal value achievable using k edges.

Moreover, we can acquire a tight *online* data-dependent bound on the solution quality:

Theorem 3 (Leskovec et al. (2007a))**.** *For a network $\hat{\mathcal{G}}$, and each edge $e \notin \hat{\mathcal{G}}$, let $\delta_e = F(\mathcal{C}|\hat{\mathcal{G}} \cup \{e\}) - F(\mathcal{C}|\hat{\mathcal{G}})$. Let $e_1, \dots e_B$ be the sequence with δ_e in decreasing order, where B is the total number of edges with marginal gain greater than 0. Then,*

$$\max_{|\mathcal{G}| \leq k} F(\mathcal{C}|\mathcal{G}) \leq F(\mathcal{C}|\hat{\mathcal{G}}) + \sum_{i=1}^{k} \delta_{e_i}.$$

Theorem 3 computes how far a given $\hat{\mathcal{G}}$ (obtained by *any* algorithm) is from the unknown NP-hard to find optimum.

Algorithm 2 The NetInf Algorithm

```
Require: Cascades $\mathcal{C} = \{\mathbf{t}^1, \ldots, \mathbf{t}^{|\mathcal{C}|}\}$, number of edges $k$
  $\mathcal{G} \leftarrow \bar{\mathcal{K}}$
  for all $c \in \mathcal{C}$ do
    $\mathcal{T}_c \leftarrow dag\_tree(c)$                {Find most likely tree (Algorithm 1)}
  end for
  while $|\mathcal{G}| < k$ do
    for all $(j,i) \notin \mathcal{G}$ do
      $\delta_{j,i} = 0$           {Marginal improvement of adding edge $(j,i)$ to $\mathcal{G}$}
      $M_{j,i} \leftarrow \emptyset$
      for all $c : t_j < t_i$ in $\mathbf{t}^c$ do
        Let $w_c(m,n)$ be the weight of $(m,n)$ in $\mathcal{G} \cup \{(j,i)\}$
        if $w_c(j,i) \geq w_c(Par_{T_c}(i), i)$ then
          $\delta_{j,i} = \delta_{j,i} + w_c(j,i) - w_c(Par_{T_c}(i), i)$
          $M_{j,i} \leftarrow M_{j,i} \cup \{c\}$
        end if
      end for
    end for
    $(j^*, i^*) \leftarrow \arg\max_{(j,i) \in \mathcal{C} \setminus \mathcal{G}} \delta_{j,i}$
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{(j^*, i^*)\}$
    for all $c \in M_{j^*,i^*}$ do
      $Par_{T_c}(i^*) \leftarrow j^*$
    end for
  end while
  return $\mathcal{G}$;
```

Speeding-up the NetInf algorithm

To make the algorithm scale to networks with thousands of nodes we speed-up the algorithm by several orders of magnitude by considering two following two improvements:

Localized update: Let C_i be the subset of cascades that go through the node i (*i.e.*, cascades in which node i is activated). Then consider that in some step n the *greedy algorithm* selects the network edge (j,i) with marginal gain $\delta_{j,i}$,and now we have to update the optimal tree of each cascade. We make a simple observation that adding the network edge (j,i) may only change the optimal trees of the cascades in the set $\mathcal{C}_i$ and thus we only need to revisit (and potentially update) the trees of cascades in $\mathcal{C}_i$.

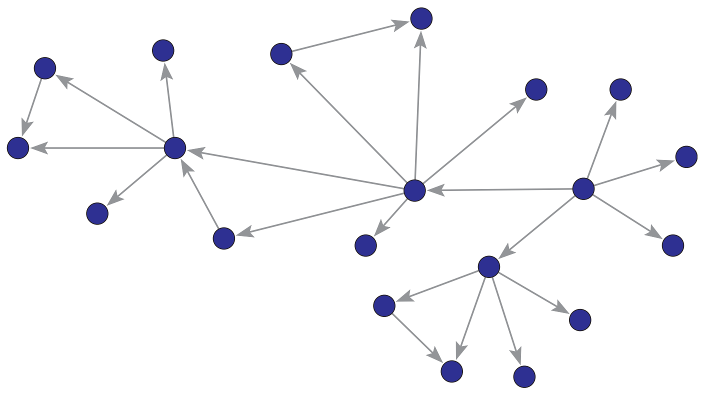
Since cascades are local (*i.e.*, each cascade hits only a relatively small subset of the network), this localized updating procedure speeds up the algorithm considerably.
Lazy evaluation: It can be used to drastically reduce the number of evaluations of marginal gains $F(\mathcal{C}|\mathcal{G}\cup\{e\})-F(\mathcal{C}|\mathcal{G})$ (Leskovec et al., 2007a). This procedure relies on the submodularity of $F(\cdot|\mathcal{G})$. The key idea behind lazy evaluations is the following. Suppose $\mathcal{G}_1, \ldots, \mathcal{G}_k$ is the sequence of networks produced during iterations of the greedy algorithm. Now let us consider the marginal gain

$$\Delta_e(\mathcal{G}_i) = F(\mathcal{C}|\mathcal{G}_i \cup \{e\}) - F(\mathcal{C}|G_i)$$

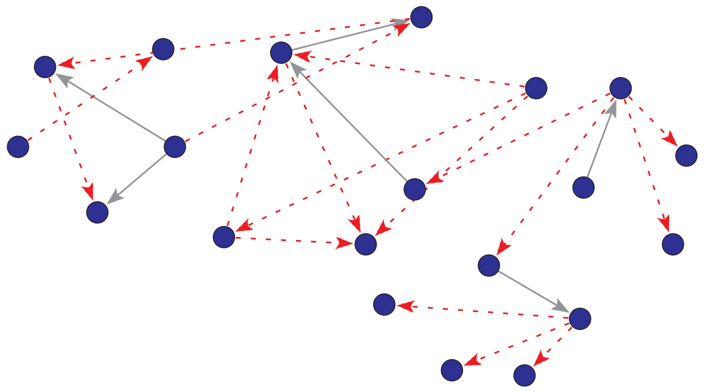
of adding some edge e to any of these networks. Due to the submodularity of $F(\cdot|\mathcal{G})$ it holds that $\Delta_e(\mathcal{G}_i) \geq \Delta_e(\mathcal{G}_j)$ whenever $i \leq j$. Thus, the marginal gains of e can only monotonically decrease over the course of the greedy algorithm. This means that elements which achieve very little marginal gain at iteration i cannot suddenly produce large marginal gain at subsequent iterations. This insight can be exploited by maintaining a priority queue data structure over the edges and their respective marginal gains. At each iteration, the greedy algorithm retrieves the highest weight (priority) edge. Since its value may have decreased from previous iterations, it recomputes its marginal benefit. If the marginal gain remains the same after recomputation, it has to be the edge with highest marginal gain, and the greedy algorithm will pick it. If it decreases, one reinserts the edge with its new weight into the priority queue and continues. Formal details and pseudo-code can be found in Leskovec et al. (2007a).

As we will show later, these two improvements decrease the run time by several orders of magnitude with *no loss* in the solution quality. We call the algorithm that implements the greedy algorithm on this alternative formulation with the above speedups the NetInf algorithm (Algorithm 2). In addition, NetInf nicely lends itself to parallelization as likelihoods of individual cascades and likelihood improvements of individual new edges can simply be computed independently. This allows us to tackle even bigger networks in shorter amounts of time.

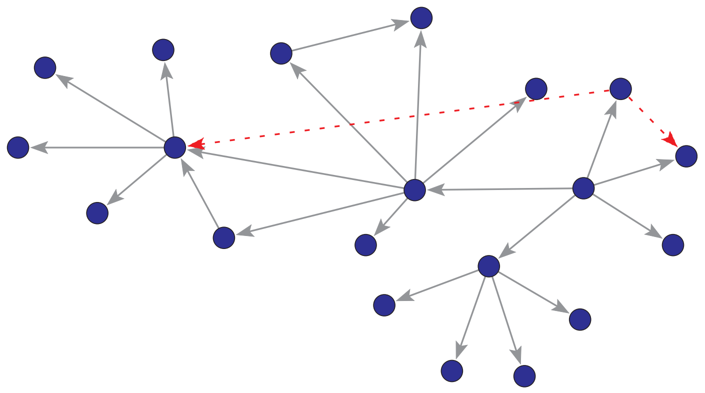
A space and runtime complexity analysis of NetInf depends heavily of the structure of the network, and therefore it is necessary to make strong assumptions on the



(a) True network G^*



(b) Inferred network $\hat{G}$ using heuristic baseline method



(c) Inferred network $\hat{G}$ using NetInf algorithm

Figure 4.4: *Diffusion network inference problem.* There is an unknown network (a) over which contagions propagate. We are given a collection of node infection times and aim to recover the network in figure (a). Using a baseline heuristic (see Section 4.3.2) we recover network (b) and using the proposed NetInf algorithm we recover network (c). Red edges denote mistakes.

structure. Due to this, it is out of the scope of this dissertation to include a formal complexity analysis. Instead, we include an empirical runtime analysis in the following section.

4.3.2 Experiments on synthetic data

The first goal of the experiments on synthetic data is to understand how the underlying network structure, the propagation model, the cascade coverage, the exogenous factors and the noise affect the performance of our algorithm. The second goal is to evaluate the effect of simplification we had to make in order to arrive to an efficient network inference algorithm. Namely, we assume the contagion propagates in a tree pattern $\mathcal{T}$ (*i.e.*, exactly $E_{\mathcal{T}}$ edges caused the propagation), consider only the most likely tree $\mathcal{T}$ (Eq. 4.8), and treat non-propagating network edges as ε-edges (Eq. 4.7).

In general, in all our experiments we proceed as follows: We are given a true diffusion network G^*, and then we simulate the propagation of a set of contagions c over the network G^* for a choice of α and β. Diffusion of each contagion creates a cascade and for each cascade, we record the node activation times t_i. Then, given these node hit times, we aim to recover the network G^* using the NetInf algorithm. For example, Figure 4.4(a) shows a network G^* of 20 nodes and 23 directed edges. Using the exponential transmission time model with $\alpha = 1$ and $\beta = 0.2$ we generated 24 cascades. Now given the node infection times, we aim to recover G^*. A baseline method (b) (described below) performed poorly while NetInf (c) recovered G^* almost perfectly by making only two errors (red edges).

Experimental setup

Our experimental methodology is composed of the following steps:

1. Ground truth network G^*.

2. Cascade generation: Probability of propagation β, and the transmission time model with parameter α.

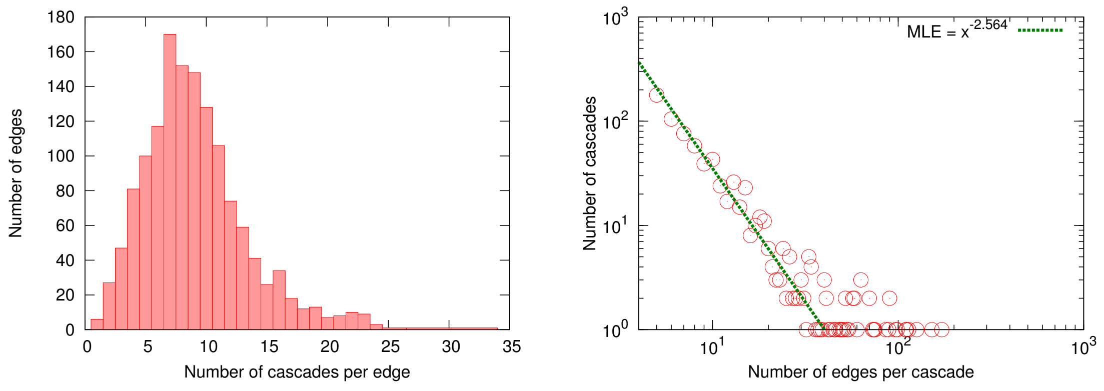
3. Number of cascades.

(1) Ground truth network G^*: We consider two models of directed real-world networks to generate G^*, namely, the Forest Fire model (Leskovec et al., 2005) and the Kronecker Graphs model Leskovec et al. (2010). For Kronecker networks, we consider three sets of parameters that produce networks with a very different global network structure: a random network (Erdős and Rényi, 1960), a core-periphery network (Leskovec et al., 2008) and a network with hierarchical community structure (Clauset et al., 2008). A brief introduction to Forest Fire and Kronecker Graph models can be found in Section 2.2.2.

(2) Cascade generation: We then simulate cascades on G^* using the generative model defined in Section **??**. For the simulation we need to choose the transmission time model (*i.e.*, pairwise transmission likelihood and parameter α). We also need to fix the parameter β, that controls probability of a cascade propagating over an edge. Intuitively, α controls how fast the cascade spreads (*i.e.*, how long the transmission times are), while β controls the size of the cascades. Large β means cascades will likely be large, while small β makes most of the edges fail to transmit the contagion which results in small infections.

(3) Number of cascades: Intuitively, the more data our algorithm gets the more accurately it should infer $\mathcal{G}^*$. To quantify the amount of data (number of different cascades) we define $\mathcal{E}_l$ to be the set of edges that participate in at least l cascades. This means $\mathcal{E}_l$ is a set of edges that transmitted at least l contagions. It is important to note that if an edge of G^* did not participate in any cascade (*i.e.*, it never transmitted a contagion) then there is no trace of it in our data and thus we have no chance to infer it. In our experiments we choose the minimal amount of data (*i.e.*, $l = 1$) so that we at least in principle could infer the true network $\mathcal{G}^*$. Thus, we generate as many cascades as needed to have a set $\mathcal{E}_l$ that contains a fraction f of all the edges of the true network $\mathcal{G}^*$. In all our experiments we pick cascade starting nodes uniformly at random and generate enough cascades so that 99% of the edges in $\mathcal{G}^*$ participate in at least one cascade, *i.e.*, 99% of the edges are included in $\mathcal{E}_1$.

Table 4.2 shows experimental values of number of cascades that let $\mathcal{E}_1$ cover different percentages of the edges. To have a closer look at the cascade size distribution, for a Forest Fire network on 1,024 nodes and 1,477 edges, we generated 4,038 cascades.



(a) FF: Cascades per edge (b) FF: Cascade size

Figure 4.5: Number of cascades per edge and cascade sizes for a Forest Fire network (1,024 nodes, 1,477 edges) with forward burning probability 0.20, backward burning probability 0.17 and exponential transmission time model with parameter $\alpha = 1$ and propagation probability $\beta = 0.5$. The cascade size distribution follows a power-law. We found the power-law coefficient using maximum likelihood estimation (MLE).

The majority of edges took part in 4 to 12 cascades and the cascade size distribution follows a power law (Figure 4.5(b)). The average and median number of cascades per edge are 9.1 and 8, respectively (Figure 4.5(a)).

Baseline method

To infer a diffusion network $\hat{\mathcal{G}}$, we consider the a simple baseline heuristic where we compute the score of each edge and then pick k edges with highest score.

More precisely, for each *possible* edge (i, j) of $\mathcal{G}$, we compute $w(i, j) = \sum_{c \in C} f(t_j | t_i; \alpha)$, *i.e.*, overall how likely were the cascades $\mathbf{t}^c \in \mathcal{C}$ to propagate over the edge (i, j). Then we simply pick the k edges (i, j) with the highest score $w(i, j)$ to obtain $\hat{\mathcal{G}}$. For example, Figure 4.4(b) shows the results of the baseline method on a small network.

Solution quality

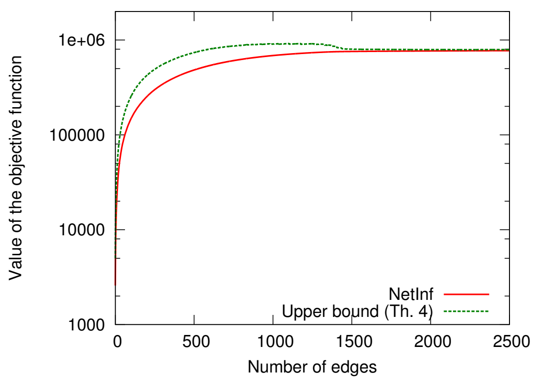
We evaluate the performance of the NetInf algorithm in two different ways. First, we are interested in how successful NetInf is at optimizing the objective function $F(C|G)$ that is NP-hard to optimize exactly. Using the online bound in Theorem 3,

Type of network	f	|C|	r	BEP	AUC
Forest Fire	0.5	388	2,898	0.393	0.29
	0.9	2,017	14,027	0.75	0.67
	0.95	2,717	19,418	0.82	0.74
	0.99	4,038	28,663	0.92	0.86
Hierarchical Kronecker	0.5	289	1,341	0.37	0.30
	0.9	1,209	5,502	0.81	0.80
	0.95	1,972	9,391	0.90	0.90
	0.99	5,078	25,643	0.98	0.98
Core-periphery Kronecker	0.5	140	1,392	0.31	0.23
	0.9	884	9,498	0.84	0.80
	0.95	1,506	14,125	0.93	0.91
	0.99	3,110	30,453	0.98	0.96
Flat Kronecker	0.5	200	1,324	0.34	0.26
	0.9	1,303	7,707	0.84	0.83
	0.95	1,704	9,749	0.89	0.88
	0.99	3,652	21,153	0.97	0.97

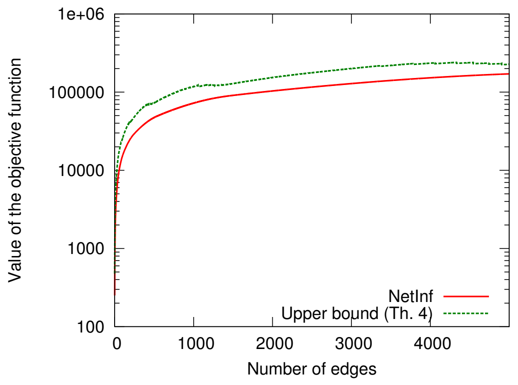
Table 4.2: Performance of synthetic data. Break-even Point (BEP) and Receiver Operating Characteristic (AUC) when we generated the minimum number of $|\mathcal{C}|$ cascades so that f-fraction of edges participated in at least one cascades $|\mathcal{E}_l| \geq f|\mathcal{E}|$. These $|\mathcal{C}|$ cascades generated the total of r edge transmissions, *i.e.*, average cascade size is $r/|\mathcal{C}|$. All networks have 1,024 nodes and 1,446 edges. We use the exponential transmission time model with parameter $\alpha = 1$, and in each case we set the probability β such that $r/|C|$ is neither too small nor too large (*i.e.*, $\beta \in (0.1, 0.6)$).

we can assess at most how far from the unknown optimal the NetInf solution is in terms of the log-likelihood score. Second, we also evaluate the NetInf based on accuracy, *i.e.*, what fraction of edges of $\mathcal{G}^*$ NetInf managed to infer correctly.

Figure 4.6(a) plots the value of the log-likelihood improvement $F(C|\mathcal{G})$ as a function of the number of edges in $\mathcal{G}$. In red we plot the value achieved by NetInf and in green the upper bound using Theorem 3. The plot shows that the value of the unknown optimal solution (that is NP-hard to compute exactly) is somewhere between the red and the green curve. Notice that the band between two curves, the optimal and the NetInf curve, is narrow. For example, at 2,000 edges in $\hat{\mathcal{G}}$, NetInf finds the solution that is least 97% of the optimal network. Moreover, also notice a strong diminishing return effect. The value of the objective function flattens out



(a) Kronecker network



(b) Real MemeTracker data

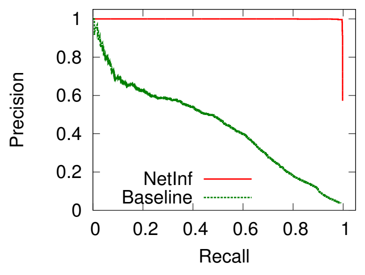
Figure 4.6: Score achieved by NetInf in comparison with the online upper bound from Theorem 3. In practice NetInf finds networks that are at 97% of NP-hard to compute optimal.

after about 1,000 edges. This means that, in practice, very sparse solutions (almost tree-like diffusion networks) already achieve very high values of the objective function close to the optimal.

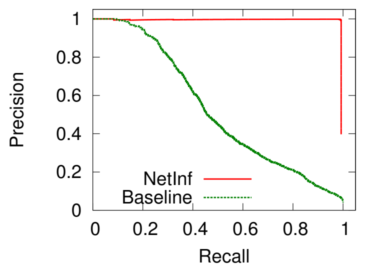
Accuracy of NetInf

We also evaluate our approach by studying how many edges inferred by NetInf are actually present in the true network $\mathcal{G}^*$. We measure the precision and recall of our method. For every value of k ($1 \leq k \leq n(n-1)$) we generate $\hat{\mathcal{G}}_k$ on k edges by using NetInf or the baseline method. We then compute precision (which fraction of edges in $\hat{\mathcal{G}}_k$ is also present $\mathcal{G}^*$) and recall (which fraction of edges of $\mathcal{G}^*$ appears in $\hat{\mathcal{G}}_k$). For small k, we expect low recall and high precision as we select the few edges that we are the most confident in. As k increases, precision will generally start to drop but the recall will increase.

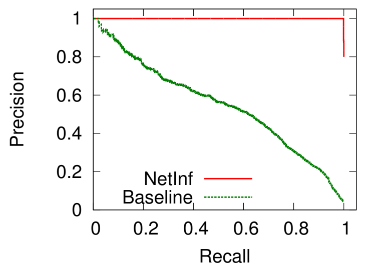
Figure 4.7 shows the precision-recall curves of NetInf and the baseline method on three different Kronecker networks (random, hierarchical community structure and core-periphery structure) with 1024 nodes and two transmission time models. The cascades were generated with an exponential transmission time model with $\alpha = 1$, or a power law transmission time model with $\alpha = 2$ and a value of β low enough to



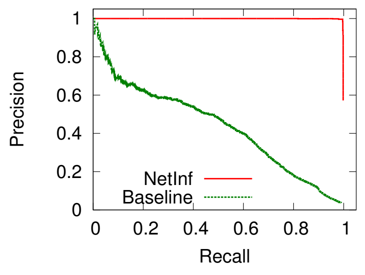
(a) Hier. Kronecker (Exp)



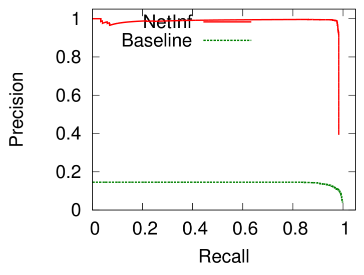
(b) Core-Periph. Kronecker (Exp)



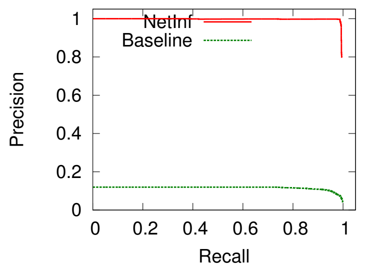
(c) Flat Kronecker (Exp)



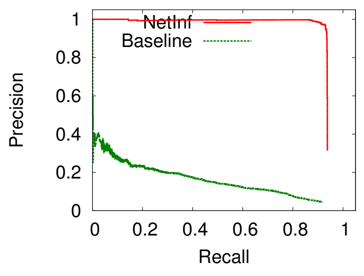
(d) Hier. Kronecker (Pow)



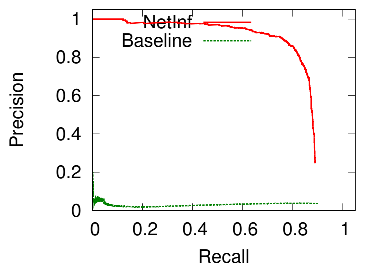
(e) Core-Periph. Kronecker (Pow)



(f) Flat Kronecker (Pow)



(g) Forest Fire (Pow, $\alpha = 1.1$)



(h) Forest Fire (Pow, $\alpha = 3$)

Figure 4.7: Precision and recall for three 1024 node Kronecker and Forest Fire network networks with exponential (Exp) and power law (Pow) transmission time model. The plots are generated by sweeping over values of k, that controls the sparsity of the solution.

avoid generating too large cascades (in all cases, we pick a value of $\beta \in (0.1, 0.6)$). For each network we generated between 2,000 and 4,000 cascades so that 99% of the edges of G^* participated in at least one cascade. We chose cascade starting points uniformly at random.

First, we focus on Figures 4.7(a), 4.7(b) and 4.7(c) where we use the exponential transmission time model on different Kronecker networks. Notice that the baseline method achieves the break-even point[1] between 0.4 and 0.5 on all three networks. On the other hand, NetInf performs much better with the break-even point of 0.99 on all three datasets.

We view this as a particularly strong result as we were especially careful not to generate too many cascades since more cascades mean more evidence that makes the problem easier. Thus, using a very small number of cascades, where every edge of $\mathcal{G}^*$ participates in only a few cascades, we can almost perfectly recover the underlying diffusion network $\mathcal{G}^*$. Second important point to notice is that the performance of NetInf seems to be strong regardless of the structure of the network $\mathcal{G}^*$. This means that NetInf works reliably regardless of the particular structure of the network of which contagions propagated (refer to Table 4.2).

Similarly, Figures 4.7(d), 4.7(e) and 4.7(f) show the performance on the same three networks but using the power law transmission time model. The performance of the baseline now dramatically drops. This is likely due to the fact that the variance of power-law (and heavy tailed distributions in general) is much larger than the variance of an exponential distribution. Thus the diffusion network inference problem is much harder in this case. As the baseline pays high price due to the increase in variance with the break-even point dropping below 0.1 the performance of NetInf remains stable with the break even point in the high 90s.

We also examine the results on the Forest Fire network (Figures 4.7(g) and 4.7(h)). Again, the performance of the baseline is very low while NetInf achieves the break-even point at around 0.90.

Generally, the performance on the Forest Fire network is a bit lower than on the Kronecker networks. However, it is important to note that while these networks have

[1]The point at which recall is equal to precision.

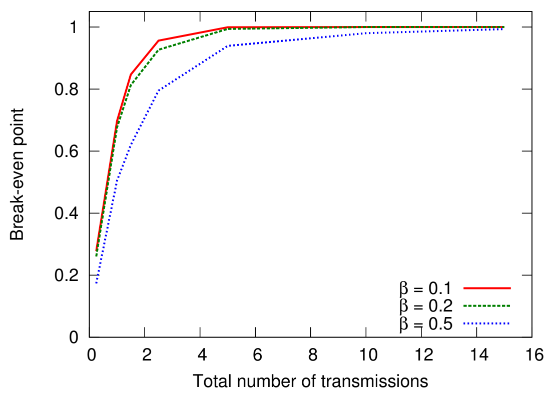


Figure 4.8: Performance of NetInf as a function of the amount of cascade data. The units in the x-axis are normalized. $x = 1$ means that the total number of transmission events used for the experiment was equal to the number of edges in $\mathcal{G}^*$. On average NetInf requires about two propagation events per edge of the original network in order to reliably recover the true network structure.

very different global network structure (from hierarchical, random, scale free to core periphery) the performance of NetInf is remarkably stable and does not seem to depend on the structure of the network we are trying to infer or the particular type of cascade transmission time model.

Finally, in all the experiments, we observe a sharp drop in precision for high values of recall (near 1). This happens because the greedy algorithm starts to choose edges with low marginal gains that may be false edges, increasing the probability to make mistakes.

Performance vs. cascade coverage

Intuitively, the larger the number of cascades that spread over a particular edge the easier it is to identify it. On one hand if the edge never transmitted then we can not identify it, and the more times it participated in the transmission of a contagion the easier should the edge be to identify.

In our experiments so far, we generated a relatively small number of cascades. Next, we examine how the performance of NetInf depends on the amount of available cascade data. This is important because in many real world situations the data of

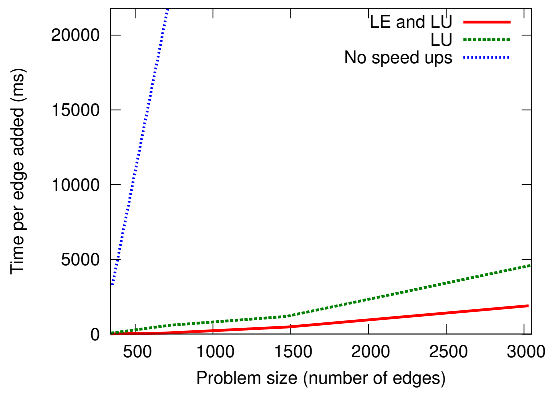


Figure 4.9: Average time per edge added by our algorithm implemented with lazy evaluation (LE) and localized update (LU).

only a few different cascades is available.

Figure 4.8 plots the break-even point of NetInf as a function of the available cascade data measured in the number of contagion transmission events over all cascades. The total number of contagion transmission events is simply the sum of cascade sizes. Thus, $x = 1$ means that the total number of transmission events used for the experiment was equal to the number of edges in $\mathcal{G}^*$. Notice that as the amount of cascade data increases the performance of NetInf also increases. Overall we notice that NetInf requires a total number of transmission events to be about 2 times the number of edges in $\mathcal{G}^*$ to successfully recover most of the edges of $\mathcal{G}^*$.

Moreover, the plot shows the performance for different values of edge transmission probability β. As noted before, big values of β produce larger cascades. Interestingly, when cascades are small (small β) NetInf needs less data to infer the network than when cascades are larger. This occurs because the larger a cascade, the more difficult is to infer the parent of each node, since we have more potential parents for each the node to choose from. For example, when $\beta = 0.1$ NetInf needs about $2|E|$ transmission events, while when $\beta = 0.5$ it needs twice as much data (about $4|E|$ transmissions) to obtain the break even point of 0.9.

Stopping criterion

In practice one does not know how long to run the algorithm and how many edges to insert into the network $\hat{\mathcal{G}}$. Given the results from Figure 4.6, we found the following heuristic to give good results. We run the NetInf algorithm for k steps where k is chosen such that the objective function is "close" to the upper bound, *i.e.*, $F(C|\hat{\mathcal{G}}) > x \cdot \text{OPT}$, where OPT is obtained using the online bound. In practice we use values of x in range 0.8–0.9. That means that in each iteration k, OPT is computed by evaluating the right hand side expression of the equation in Theorem 3, where k is simply the iteration number. Therefore, OPT is computed online, and thus the stopping condition is also updated online.

Scalability

Figure 4.9 shows the average computation time per edge added for the NetInf algorithm implemented with lazy evaluation and localized update. We use a hierarchical Kronecker network and an exponential transmission time model with $\alpha = 1$ and $\beta = 0.5$. Localized update speeds up the algorithm for an order of magnitude ($45\times$) and lazy evaluation further gives a factor of 6 improvement. Thus, overall, we achieve two orders of magnitude speed up ($280\times$), without *any* loss in solution quality.

In practice the NetInf algorithm can easily be used to infer networks of 10,000 nodes in a matter of hours.

Performance vs. transmission time noise

In our experiments so far, we have assumed that the transmission time values between infections are not *noisy* and that we have access to the true distribution from which the transmission times are drawn. However, real data may violate any of these two assumptions.

We study the performance of NetInf (break-even point) as a function of the noise of the waiting time between infections. Thus, we add Gaussian noise to the waiting times between infections in the cascade generation process.

Figure 4.10 plots the performance of NetInf (break-even point) as a function of

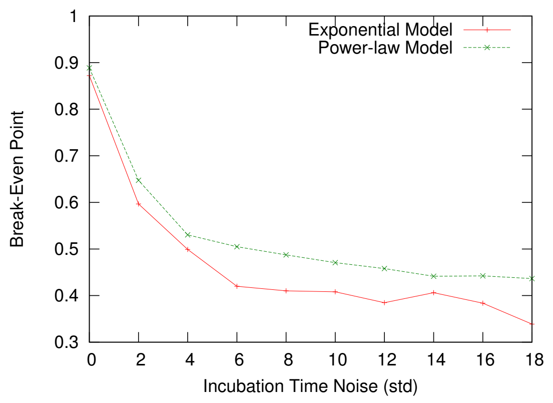


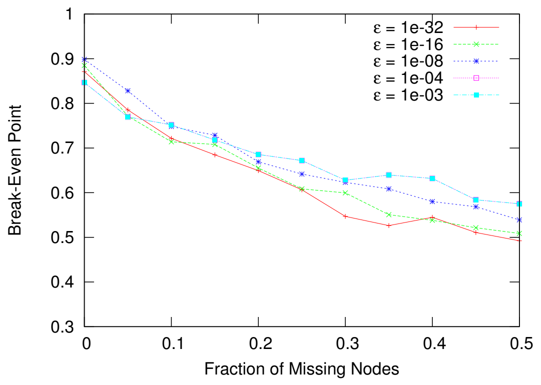
Figure 4.10: Break-even point of NetInf as a function of the amount of additive Gaussian noise in the transmission time.

the amount of Gaussian noise added to the transmission times between infections for both an exponential transmission time model with $\alpha = 1$, and a power law transmission time model with $\alpha = 2$. The break-even point degrades with noise but once a high value of noise is reached, an additional increment in the amount of noise does not degrade further the performance of NetInf. Interestingly, the break-even point value for high values of noise is very similar to the break-even point achieved later in a real dataset (Figures 4.12(a) and 4.12(b)).

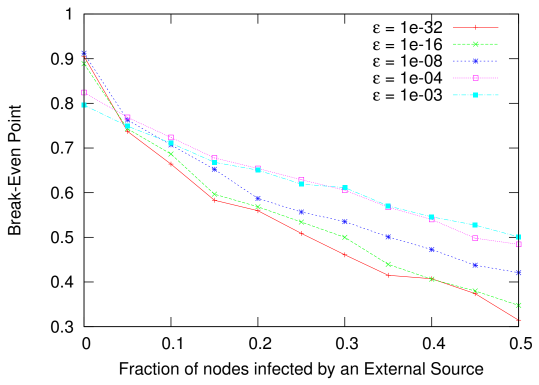
Performance vs. infections by the external source

In all our experiments so far, we have assumed that we have access to *complete* cascade data, *i.e.*, we are able to observe all the nodes taking part in each cascade. Thereby, except for the first node of a cascade, we do not have any "jumps" or missing nodes in the cascade as it spreads across the network. Even though techniques for coping with missing data in information cascades have recently been investigated (Sadikov et al., 2011), we evaluate NetInf against both scenarios.

First, we consider the case where a random fraction of each cascade is missing. This means that we first generate a set of cascades, but then only record node infection times of f-fraction of nodes. We first generate enough cascades so that without counting the missing nodes in the cascades, we still have that 99% of the edges in G^*



(a) Missing node infection data



(b) Node infections due to external source

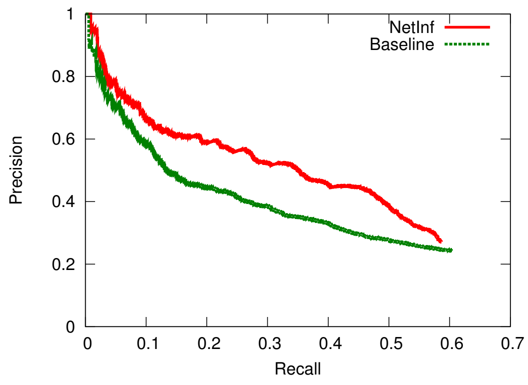
Figure 4.11: Break-even point of NetInf as (a) function of the fraction of missing nodes per cascade, and as (b) function of the fraction of nodes that are activated by an external source per cascade.

participate in at least one cascade. Then we randomly delete (*i.e.*, set infection times to infinity) f-fraction of nodes in each cascade.

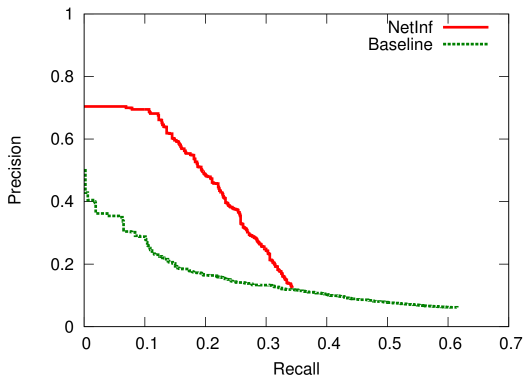
Figure 4.11(a) plots the performance of NetInf (break-even point) as a function of the percentage of missing nodes in each cascade. Naturally, the performance drops with the amount of missing data. However, we also note that the effect of missing nodes can be mitigated by an appropriate choice of the parameter ε. Basically, higher ε makes propagation via ε-edges more likely and thus by giving a cascade a greater chance to propagate over the ε-edges NetInf can implicitly account for the missing data.

Second, we also consider the case where the contagion does not spread through the network via diffusion but rather due to the influence of an external source. Thus, the contagion does not really spread over the edges of the network but rather appears almost at random at various nodes of the network.

Figure 4.11(b) plots the performance of NetInf (break-even point) as a function of the percentage of nodes that are activated by an external source for different values of ε. In our framework, we model the influence due to the external source with the ε-edges. Note that appropriately setting ε can appropriately account for the exogenous infections that are not the result of the network diffusion but due to the



(a) Hyperlinks cascades



(b) Memetracker cascades

Figure 4.12: Precision and recall for a 500 node hyperlink network using (a) hyperlinks cascades and (b) MemeTracker cascades.

external influence. The higher the value of ε, the stronger the influence of the external source, *i.e.*, we assume a greater number of missing nodes or number of nodes that are activated by an external source. Thus, the break-even is more robust for higher values of ε.

4.3.3 Experiments on real data

Dataset description

We use the original MemeTracker dataset, which contains more than 172 million news articles and blog posts from 1 million online sources over a period of one year from September 1 2008 till August 31 2009[2]. Based on this raw data, we use two different methodologies to trace information on the Web and then create two different datasets: hyperlink cascade dataset and MemeTracker cascade dataset. We refer the reader to Section 2.5.2 for more details on the MemeTracker dataset and an in-depth description of the two methodologies we used to trace information on the Web.

[2]Data available at `http://memetracker.org` and `http://snap.stanford.edu/netinf`

Accuracy on real data

As there is not ground truth network for both datasets, we use the following way to create the ground truth network G^*. We create a network where there is a directed edge (i, j) between a pair of nodes i and j if a post on site i linked to a post on site j. To construct the network we take the top 500 sites in terms of number of hyperlinks they create/receive. We represent each site as a node in $\mathcal{G}^*$ and connect a pair of nodes if a post in first site linked to a post in the second site. This process produces a ground truth network $\mathcal{G}^*$ with 500 nodes and 4,000 edges.

First, we use the blog hyperlink cascades dataset to infer the network $\hat{\mathcal{G}}$ and evaluate how many edges NetInf got right. Figure 4.12(a) shows the performance of NetInf and the baseline. Notice that the baseline method achieves the break-even point of 0.34, while our method performs better with a break-even point of 0.44, almost a 30% improvement.

NetInf is basically performing a link-prediction task based only on temporal linking information. The assumption in this experiment is that sites prefer to create links to sites that recently mentioned information while completely ignoring the authority of the site. Given such assumption is not satisfied in real-life, we consider the break even point of 0.44 a good result.

Now, we consider an even harder problem, where we use the Memetracker dataset to infer $\mathcal{G}^*$. In this experiment, we only observe times when sites mention particular textual phrases and the task is to infer the hyperlink structure of the underlying web graph. Figure 4.12(b) shows the performance of NetInf and the baseline. The baseline method has a break-even point of 0.17 and NetInf achieves a break-even point of 0.28, more than a 50% improvement

To have a fair comparison with the synthetic cases, notice that the exponential transmission time model is a simplistic assumption for our real dataset, and NetInf can potentially gain additional accuracy by choosing a more realistic transmission time model.

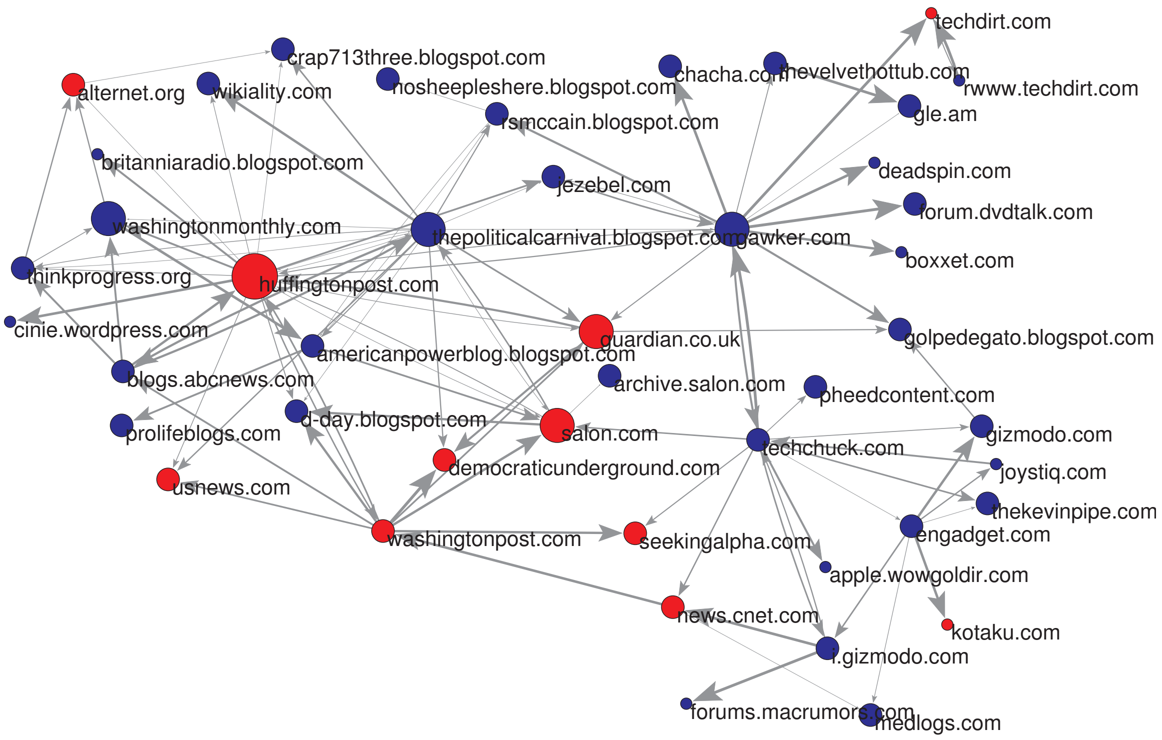


Figure 4.13: Small part of a news media (red) and blog (blue) diffusion network. We use the blog hyperlink cascades dataset, *i.e.*, hyperlinks between blog and news media posts to trace the flow of information.

Solution quality

Similarly as with synthetic data, in Figure 4.6(b) we investigate the value of the objective function and compare it to the online bound. Notice that the bound is almost as tight as in the case of synthetic networks, finding the solution that is least 84% of optimal and both curves are similar in shape to the synthetic case value. Again, as in the synthetic case, the value of the objective function quickly flattens out which means that one needs a relatively few number of edges to capture most of the information flow on the Web.

In the remainder of the section, we use the top 1,000 media sites and blogs with the largest number of documents.

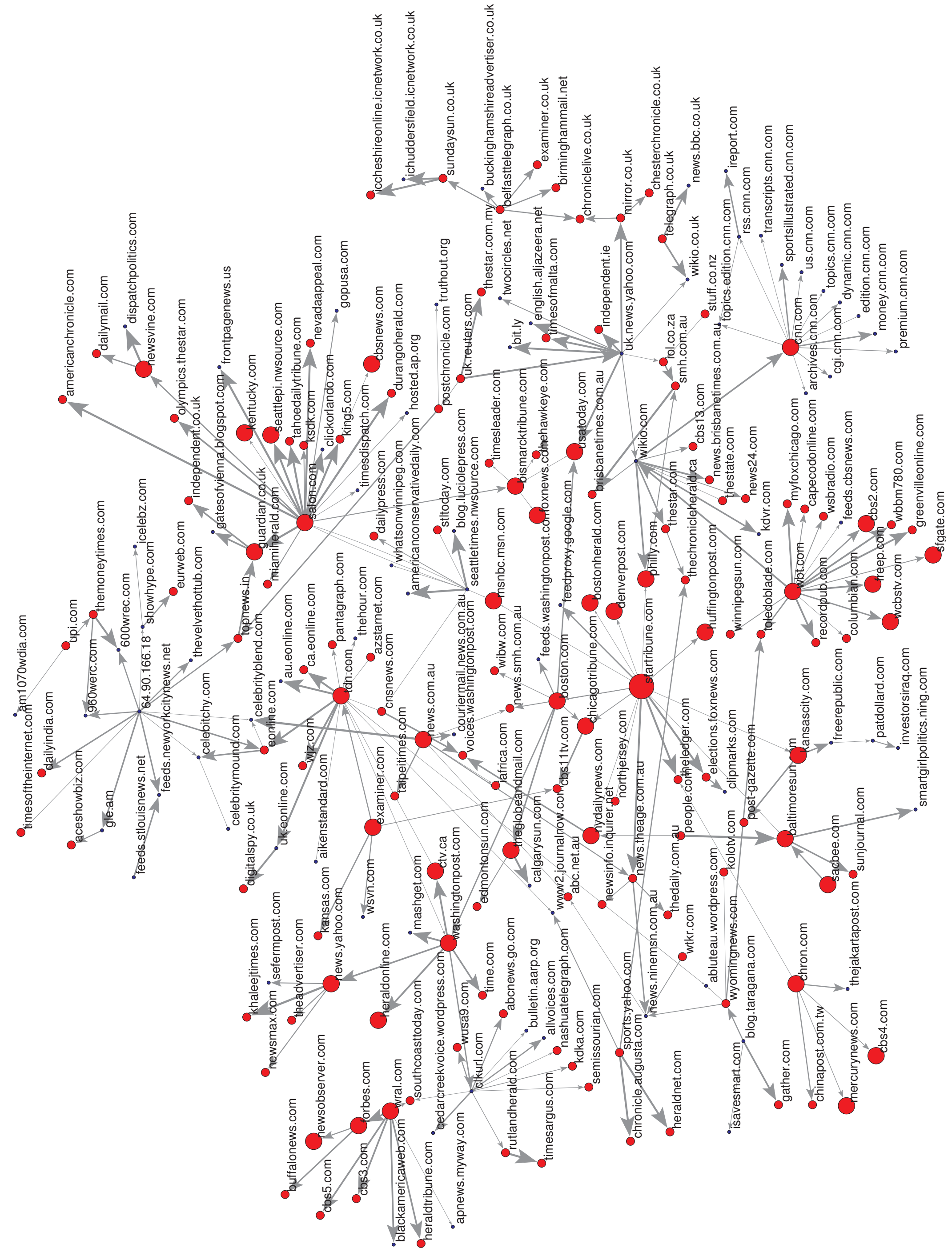


Figure 4.14: Small part of a news media (red) and blog (blue) diffusion network. We use the MemeTracker dataset, *i.e.*, textual phrases from MemeTracker to trace the flow of information.

Visualization of diffusion networks

We examine the structure of the inferred diffusion networks using both datasets: the blog hyperlink cascades dataset and the MemeTracker dataset.

Figure 4.13 shows the largest connected component of the diffusion network after 100 edges have been chosen using the first dataset, *i.e.*, using hyperlinks to track the flow of information. The size of the nodes is proportional to the number of articles on the site and the width of the edge is proportional to the probability of influence, *i.e.*, stronger edges have higher width. The strength of an edge across all cascades is simply defined as the marginal gain given by adding the edge in the greedy algorithm (and this is proportional to the probability of influence). Since news media articles rarely use hyperlinks to refer to one another, the network is somewhat biased towards web blogs (blue nodes). There are several interesting patterns to observe.

First, notice that three main clusters emerge: on the left side of the network we can see blogs and news media sites related to politics, at the right top, we have blogs devoted to gossip, celebrity news or entertainment and on the right bottom, we can distinguish blogs and news media sites that deal with technological news. As Huffington Post and Political Carnival play the central role on the political side of the network, mainstream media sites like Washington Post, Guardian and the professional blog Salon.com play the role of connectors between the different parts of the network. The celebrity gossip part of the network is dominated by the blog Gawker and technology news gather around blogs Gizmodo and Engadget, with CNet and TechChuck establishing the connection to the rest of the network.

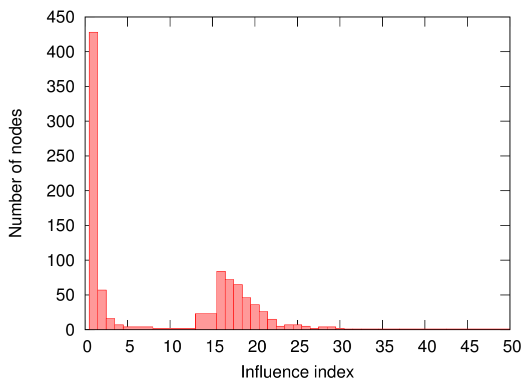
Figure 4.14 shows the largest connected component of the diffusion network after 300 edges have been chosen using the second methodology, *i.e.* using short textual phrases to track the flow of information. In this case, the network is biased towards news media sites due to its higher volume of information.

Insights into the diffusion on the web. The inferred diffusion networks also allow for analysis of the global structure of information propagation on the Web. For this analysis, we use the MemeTracker dataset and analyze the structure of the inferred information diffusion network.

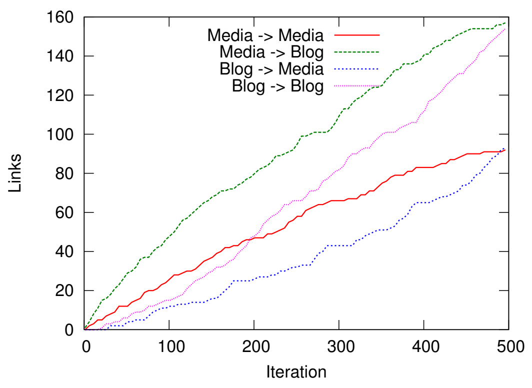
First, Figure 4.15(a) shows the distribution of the influence index. The influence index is defined as the number of reachable nodes from w by traversing edges of the inferred diffusion network (while respecting edge directions). Nevertheless, we are also interested in the distance from w to its reachable nodes, i.e. nodes at shorter distances are more likely to be activated by w. Thus, we slightly modify the definition of influence index to be $\sum_u 1/d(w,u)$ where we sum over all the reachable nodes from w and $d(w,u)$ is the distance between w and u. Notice that we have two types of nodes. There is a small set of nodes that can reach many other nodes, which means they either directly or indirectly propagate information to them. On the other side we have a large number of sites that only get influenced but do not influence many other sites. This hints at a core periphery structure of the diffusion network with a small set of sites directly or indirectly spreading the information in the rest of the network.

Figure 4.15(b) investigates the number of links in the inferred network that point between different types of sites. Here we split the sites into mainstream media and blogs. Notice that most of the links point from news media to blogs, which says that most of the time information propagates from the mainstream media to blogs. Then notice how at first many media-to-media links are chosen but in later iterations the increase of these links starts to slow down. This means that media-to-media links tend to be the strongest and NetInf picks them early. The opposite seems to occur in case of blog-to-blog links where relatively few are chosen first but later the algorithm picks more of them. Lastly, links capturing the influence of blogs on mainstream media are the rarest and weakest. This suggests that most information travels from mass media to blogs.

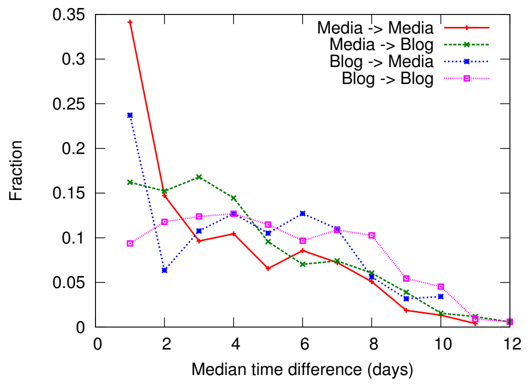
Last, Figure 4.15(c) shows the median time difference between mentions of different types of sites. For every edge of the inferred diffusion network, we compute the median time needed for the information to spread from the source to the destination node. Again, we distinguish the mainstream media sites and blogs. Notice that media sites are quick to activate one another or even to get activated from blogs. However, blogs tend to be much slower in propagating information. It takes a relatively long time for them to get "activated" with information regardless whether the information



(a) Influence Index



(b) Number of edges as iterations proceed



(c) Median edge time lag

Figure 4.15: (a) Distribution of node influence index. Most nodes have very low influence (they act as sinks). (b) Number and strength of edges between different media types. Edges of news media influencing blogs are the strongest. (c) Median time lag on edges of different type.

comes from the mainstream media or the blogosphere.

Finally, we have observed that the insights into diffusion on the web using the inferred network are very similar to insights obtained by simply taking the hyperlink network. However, our aim here is to show that (i) although the quantitative results are modest in terms of precision and recall, the qualitative insights makes sense, and that (ii) it is surprising that using simply timestamps of links, we are able to draw the same qualitative insights as using the hyperlink network.

4.4 Multitree

4.4.1 Algorithm

We initially considered the optimization problem defined by Eq. (4.5) to be intractable and left as an interesting open problem (Gomez-Rodriguez et al., 2010). In this section, we 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. (4.4), we need to compute Eq. (4.8) for each cascade $\mathbf{t}^c$. In other words, for each cascade $\mathbf{t}^c$, we need to compute a sum of likelihoods over the set $\mathcal{T}_c(\mathcal{G})$ of all possible connected spanning trees induced by the nodes activated in the 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 4 (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,*

$$(-1)^{x+y} \det(A_{x,y}) = \sum_{T \in \mathcal{T}(W)} \prod_{(i,j) \in T} w_{i,j}, \tag{4.14}$$

where $\mathcal{T}$ is each directed spanning tree in W that starts in y.

In our case, we compute Eq. (4.8) by setting $w_{i,j}$ to $f(t_j|t_i;\alpha)$ and computing the determinant in Eq. (4.14). We then compute Eq. (4.4) by multiplying the determinants of $|\mathcal{C}|$ 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 4, by construction, is also upper triangular. Fortunately, the determinant of an upper triangular matrix

Algorithm 3 The MULTITREE algorithm

```
Require: $\mathcal{C}, k$
  $\mathcal{G} \leftarrow \bar{\mathcal{K}}$;
  while $|\mathcal{G}| < k$ do
    for all $(j,i) \notin \mathcal{G} : \exists \mathbf{t}^c \in \mathcal{C}$ 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 $\mathcal{G} \cup \{(j,i)\}$;
        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
    $(j^*, i^*) \leftarrow \arg\max_{(j,i) \notin \mathcal{G}} \delta_{j,i}$;
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{(j^*, i^*)\}$;
  end while
  return G;
```

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. 4.4 in time $O(|\mathcal{C}| \cdot N^2)$, 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 $|\mathcal{C}|$ cascades.

Following a similar reasoning to NETINF, in Section 4.3.1, we introduce the concept of ε-edges to account for the fact that nodes may get activated for reasons other than the network influence. For example, in online media, not all the information propagates via the network, as some is also pushed onto the network by the mass media (Katz and Lazarsfeld, 1955; Watts and Dodds, 2007) and thus a disconnected cascade can be created. Similarly, in viral marketing, a person may purchase a product due to the influence of peers (*i.e.*, network effect) or for some other reason (*e.g.*, seing a commercial on TV) (Leskovec et al., 2006a). As noted before, adding ε-edges 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 assumed to be activated by an external source.

We then define the improvement of log-likelihood for cascade $\mathbf{t}^c$ under network $\mathcal{G}$ over an empty network $\bar{\mathcal{K}}$ (*i.e.*, network with only ε-edges and no network edges):

$$F(\mathbf{t}^c|\mathcal{G}) = \sum_{t_j \in \mathbf{t}^c} \log \left(\sum_{t_i \in \mathbf{t}^c : t_i \leq t_j} w_c(i,j) \right), \tag{4.15}$$

where $w_c(i,j) = \varepsilon^{-1} f(t_j|t_i;\alpha) \geq 0$ and $\sum_{i \in G: t_j \geq t_i} w_c(i,j) \geq 1$. Finally, maximizing Eq. (4.5) is equivalent to maximizing the following objective function:

$$F(\mathcal{C}|\mathcal{G}) = F(\mathbf{t}^1, \ldots, \mathbf{t}^{|\mathcal{C}|}|\mathcal{G}) = \sum_{\mathbf{t}^c \in \mathcal{C}} F(\mathbf{t}^c|\mathcal{G}), \tag{4.16}$$

where $\mathcal{G}$ is the variable.

Efficient optimization. By construction, the empty network $\bar{\mathcal{K}}$ has score 0, $F(\mathcal{C}|\bar{\mathcal{K}}) = 0$, and the objective function $F(\cdot|\mathcal{G})$ is non-negative monotonic, $F(\mathcal{C}|\mathcal{G}) \leq F(\mathcal{C}|\mathcal{G}')$, for any $\mathcal{G} \subseteq \mathcal{G}'$. Therefore, adding more edges to $\mathcal{G}$ never decreases the solution quality, and thus the complete network maximizes $F(\cdot|\mathcal{G})$. However, in real-world scenarios, we are interested in inferring sparse networks with a small number of edges. Thus, we would like to solve:

$$\mathcal{G}^* = \underset{|\mathcal{G}| \leq k}{\operatorname{argmax}} \, F(\mathcal{C}|\mathcal{G}), \tag{4.17}$$

where the maximization is over all directed networks $\mathcal{G}$ of at most k edges. Naively searching over all k edge networks would take time exponential in k, which is intractable. Moreover, finding the optimal solution to Eq. 4.17 is NP-hard:

Theorem 5. *The diffusion network inference problem defined by Eq. 4.17 is NP-hard.*

Proof. By reduction from the MAX-k-COVER problem (Khuller et al., 1999), similarly to the proof of Th. 1. □

While finding the optimal solution is hard, we will now show that $F(\cdot|\mathcal{G})$ satisfies *submodularity* on the set of directed edges in $\mathcal{G}$, a natural diminishing returns property. Refer to Section 2.4 for a definition of submodularity. Fortunately, submodularity allow us to efficiently find a *provable* near-optimal solution to the optimization problem:

Theorem 6. *Let $\mathcal{V}$ be a set of nodes, and $\mathcal{C}$ be a collection of cascades hitting the nodes $\mathcal{V}$. Then $F(\mathbf{t}^1, \ldots, \mathbf{t}^{|\mathcal{C}|}|\mathcal{G})$ is a submodular function $F : 2^{\mathcal{W}} \rightarrow \mathbb{R}$ defined over subsets $\mathcal{W} \subseteq \mathcal{V} \times \mathcal{V}$ of directed edges.*

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

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

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

$$F(\mathcal{C}|\mathcal{G}) = \sum_{c \in \mathcal{C}} F(\mathbf{t}^c|\mathcal{G})$$

is submodular as well. □

We now proceed as in Section 4.3.1 and optimize $F(C|G)$ by using the *greedy algorithm*, a well-known efficient heuristic with provable performance guarantees. The algorithm starts with an empty network $\bar{\mathcal{K}}$ and it adds edges that maximize the *marginal gain* sequentially. That means, at iteration i we choose the edge

$$e_i = \underset{e \in \mathcal{G} \setminus \mathcal{G}_{i-1}}{\operatorname{argmax}} F(C|\mathcal{G}_{i-1} \cup \{e\}) - F(C|\mathcal{G}_{i-1}).$$

The algorithm stops once it has selected k edges, and returns the solution $\hat{\mathcal{G}} = \{e_1, \ldots, e_k\}$. The greedy algorithm is guaranteed to find a set $\hat{\mathcal{G}}$ 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 NetInf formulation, Multitree also allows for two speeds-up: localized updates and lazy evaluation (Algorithm 3), and we can also obtain an on-line bound based simply on the submodularity of the objective function (Leskovec et al., 2007a).

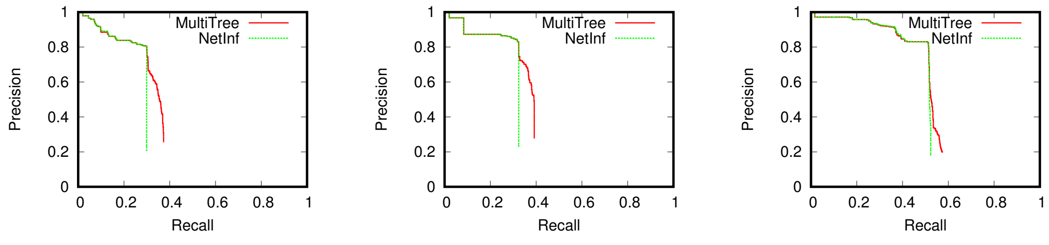
4.4.2 Experiments on synthetic data

The goal of the experiments on synthetic data is understanding to what extent Multitree benefits from considering all propagation trees per cascade, in contrast with NetInf, which considers only the most likely tree per cascade. We will show that, 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.

Experimental setup. We follow the same experimental methodology that we used in the experiments on synthetic data for NetInf, in Section 4.3.2:

1. Ground truth network G^*.
2. Cascade generation: Probability of propagation β, and the transmission time model with parameter α.
3. Number of cascades.

However, we would like to give empirical evidence that, by considering all propagation trees, Multitree is able to infer networks more accurately than NetInf when the number of observed cascades is small compared to the network size. Otherwise, their performance is surprisingly similar. Therefore, we simulate and record a relatively small set of propagating cascades over several synthetic networks, in contrast with the experimental setup in Section 4.3.2. There are several reasons why we consider small set of cascades in comparison to the network size. First, both



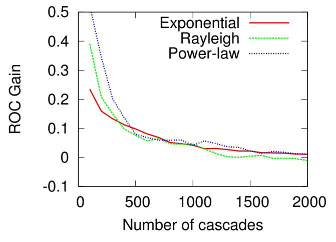
(a) Random, Exp (b) Hierarchical, Pow (c) Core-periphery, Ray

Figure 4.16: Precision against recall (PR). To control the solution sparsity or precision-recall tradeoff, we sweep over k (number of edges). (a): 1,024 node random Kronecker network with Rayleigh (Ray) model. (b): 1,024 node hierarchical Kronecker network with power-law (Pow) model. (c): 1,024 node core-periphery Kronecker network with exponential (Exp) model. In all three networks, we recorded 200 cascades.

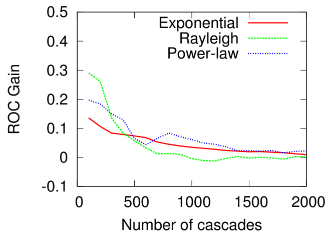
NetInf and Multitree assume that cascades propagate over a fixed network. Since social networks are highly dynamic (Backstrom and Leskovec, 2011b), changing and growing rapidly, we can only expect to record a small number of cascades over a fixed network. Second, tracking and recording cascades is a 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 two measures: precision and recall. 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 $\hat{G}$.

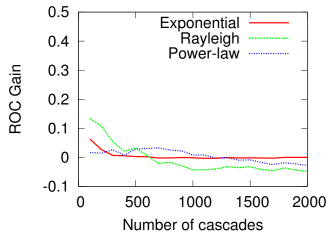
Figure 4.16 compares our method the precision, recall and accuracy of our method with for three different 1,024 node Kronecker networks: a random network (Erdős and Rényi, 1960) (parameter matrix $[0.5, 0.5; 0.5, 0.5]$), a hierarchical network (Clauset et al., 2008) ($[0.9, 0.1; 0.1, 0.9]$) and a core-periphery network (Leskovec et al., 2008) ($[0.9, 0.5; 0.5, 0.3]$), and 200 observed cascades. In terms of recall, Multitree is able to reach higher values than NetInf, *i.e.*, it is able to discover more true edges from a small number of cascades than NetInf. For recall values that are reachable using NetInf, Multitree offer a very similar precision value. Multitree allows



(a) Random



(b) Hierarchical



(c) Core-periphery

Figure 4.17: 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.

for higher recall in comparison with NetInf because it gets exhausted[3] later for considering all possible trees per cascade instead of only the most probable one.

Performance vs. cascade coverage. Intuitively, the more 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 between NetInf and Multitree become negligible. Figure 4.17 plots the gain in Area Under the ROC curve (AUC) for our method in comparison with NetInf, $(\text{AUC}_{\text{our method}} - \text{AUC}_{\text{NetInf}})/\text{AUC}_{\text{NetInf}}$, 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 Multitree 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 Multitree.

Scalability. Figure 4.18 plots the average computation time per edge added against number of cascades. Both Multitree and NetInf have a comparable performance in terms of scalability. None of them depends on the network size but the number of

[3] A greedy method gets exhausted at iteration k when there are not any more edges with marginal gain larger than zero.

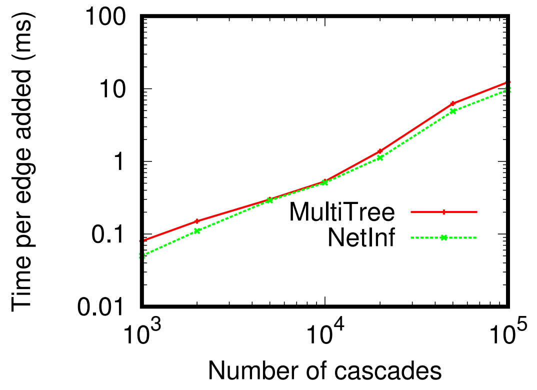


Figure 4.18: Average running time per edge added against number of cascades. We used a 1,024 node random Kronecker with exponential transmission model.

cascades and cascade size. As an experimental validation, we run our MULTITREE 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.

4.4.3 Experiments on real data

Experimental setup. We use the original MemeTracker dataset, which contains more than 172 million news articles and blog posts from 1 million online sources over a period of one year from September 1 2008 till August 31 2009[4], as in Section 4.3.3. We trace information on the Web using hyperlinks, building a hyperlink cascade dataset. We refer the reader to Section 2.5.2 for more details on the MemeTracker dataset and an in-depth description of the hyperlink cascade dataset.

In our experiments, 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 $\mathcal{G}$ 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 $\hat{\mathcal{G}}$ from the hyperlink cascades and evaluate precision, recall and accuracy with respect to G. We consider a power law pairwise transmission likelihood.

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

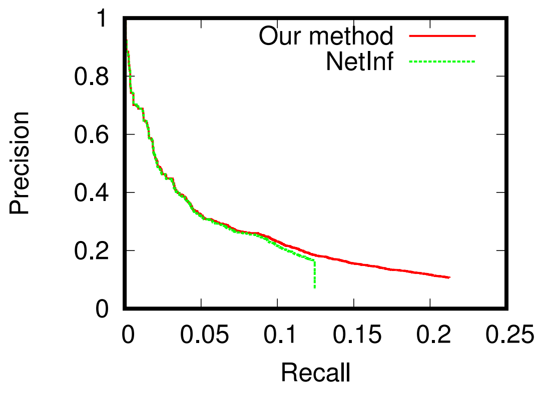


Figure 4.19: Real data. Panel (a) plots precision-recall and panel (b) accuracy on a 1,000 node hyperlink network with 10,000 edges using 1,000 cascades and a power-law model. To control the solution sparsity or precision-recall tradeoff, we sweep over k (number of edges).

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 thesis.

Accuracy. Figure 4.19 shows precision and recall of MULTITREE in comparison with NETINF. MULTITREE reaches higher recall values than NETINF, as expected, given the small number of hyperlink cascades.

4.5 Summary

In this chapter, we have formalized the problem of inferring unweighted diffusion networks, and developed two scalable algorithms: NETINF and MULTITREE. Both algorithms make minimal assumptions about the physical, biological or cognitive mechanisms responsible for diffusion. Instead, the algorithms choose *provable* sub-optimal set of k edges maximizing two different approximations to the likelihood of the observed data – temporal traces left by cascades of activations, which we show is a NP-hard problem. Qualitative assumptions about activations (*e.g.*, are they long-tailed or faddish?) determine the choice of nonparametric or parametric model on the edges. Importantly, both methods allows for the transmission likelihood to be arbitrarily complicated. This provides tremendous flexibility in fitting real data which may combine long-tailed, faddish and other qualitative behaviors.

We evaluated NetInf and Multitree on a wide range of synthetic unweighted diffusion static networks which aim to mimic the structure of real-world social and information networks. Our algorithms are able to accurately recover the underlying networks. In our experiments, NetInf and Multitree drastically outperformed a naive maximum weight baseline heuristic. Their performance is very similar, however, Multitree is able to infer networks more accurately than NetInf when the number of observed cascades is small compared to the network size.

Most importantly, our algorithms allow us to study properties of real networks in online media. We found that the inferred network exhibits a core-periphery structure with mass media influencing most of the blogosphere. Clusters of sites related to similar topics emerge (politics, gossip, technology, etc.), and a few sites with social capital interconnect these clusters allowing a potential diffusion of information among sites in different clusters.

Chapter 5

Inference of weighted diffusion networks

5.1 Introduction

This chapter presents NetRate, a method for inferring static and dynamic weighted diffusion networks based on observed activations (Gomez-Rodriguez et al., 2011), including a highly efficient implementation of NetRate, called InfoPath (Gomez-Rodriguez et al., 2013). To do so, we construct a probabilistic model incorporating some basic assumptions about the temporal structures that generate diffusion processes, as in Chapter 4. The assumptions are as follows:

A. activations are binary, *i.e.*, a node is either activated or it is not; we do not model partial activations or the partial propagation of information;

B. activations along edges of the network occur independently of each other;

C. information propagates through the network due only to diffusion, while ignoring any external sources;

D. cascades propagates independently of each other;

E. a node gets activated once the *first* parent activates the node;

F. activation along edges of the network occur at different transmission rates;

G. we observe *all* activations occurring in the network during a limited observation time window; and,

H. in the dynamic setting, we assume *unused* edges (j, i) decay exponentially §5.3.3.

Assumptions (A-D) are common to our approach to inferring unweighted diffusion networks in Chapter 4. However, assumptions (E-H) differ. Our aim is to infer not only the connectivity of the network but also the transmission rates across its edges after observing the times at which nodes in the network become activated.

In more detail, we first formulate a generative probabilistic model of diffusion that aims to realistically describe how activations occur over time in a static network. The model considers contagions to spread as directed acyclic networks (DAGs) through the network. The model considers the information which propagates through the network due only to diffusion, while ignoring any external sources (Myers et al., 2012). We then generalize the model to support dynamic networks whose structure and transmission rates changes over time. Solving both the static and dynamic network inference problem reduces to solving convex problems. The convex problem decouples into many smaller problems, allowing for natural parallelization. We develop an efficient implementation which uses stochastic gradient (Robbins and Monro, 1951) and allows us to infer networks with hundreds of thousands of nodes.

We apply our algorithm to synthetic data and to a real Web information propagation dataset of 179 million different information contagions spreading among 3.3 million blog and news media sites over a one year period, from March 2011 till February 2012, described in Section 2.5. Results on synthetic data show that our method is robust across network topologies, transmission models and variations in the transmission rate over time.

Experiments on large-scale real news and social media data lead to interesting qualitative insights and findings. For example, we find that information pathways for general recurrent topics are more stable across time than for on-going news events. Clusters of news media sites and blogs often emerge and vanish in matter of days

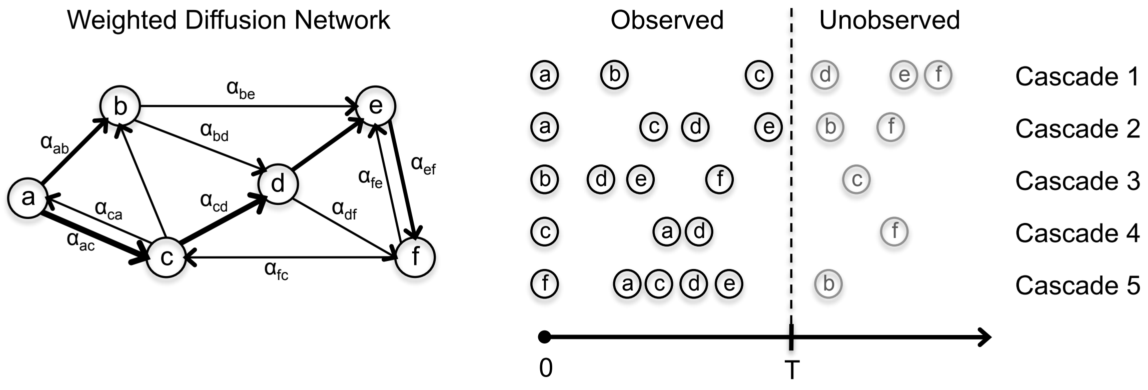


Figure 5.1: We observe a set of cascades (right) within an unknown diffusion network (left). For each contagion c, we only observe the times in which nodes get infected up to time T^c but not who infected whom. Our goal is to infer the network $\mathcal{G}$ and transmission rates $\alpha_{i,j}$ based on the observed cascades.

for on-going news events. Major social movements and events involving civil population, such as the Libyan's civil war or Syria's uprise, lead to an increased amount of information pathways among blogs as well as in the overall increase in the network centrality of blogs and social media sites.

The remainder of the chapter is organized as follows: in Section 5.2, we describe our continuous time model of diffusion over weighted static and dynamic networks and state three weighted network inference problems. In Sections 5.3, we present an efficient inference algorithm for weighted static and dynamic networks, NetRate, and evaluate it on synthetic and real diffusion data. We conclude with a summary of our results in Section 5.4.

5.2 Problem formulation

In this section, we first describe the diffusion data our inference algorithm for weighted diffusion networks is designed for and continue describing the generative model of diffusion. We conclude with a statement of the weighted network inference problem for static and dynamic networks.

Symbol	Description
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Directed network with node set V and edge set E
$(\mathcal{G}, \mathbf{A})$	Diffusion network: directed network $\mathcal{G}$ and transmission rates $\mathbf{A}$
$\mathbf{A} = [\alpha_{i,j}]$	Pairwise transmission rates for all pair of nodes (i, j)
$\alpha_{i,j}$	Pairwise transmission rate of edge (i, j)
c	Contagion
$\mathbf{t}^c$	Cascade: activation times for contagion c
$\mathcal{C}$	Set of all recorded cascades
$\mathcal{C}_t$	Set of recorded cascades by time t
t_i^c	Activation time of node i in cascade $\mathbf{t}^c$
T^c	Observation window cut-off or time horizon for cascade $\mathbf{t}^c$
$\mathbf{t}^{\leq T^c}$	Observed activation times for cascade $\mathbf{t}^c c$ up to T^c
$f(t_j\|t_i, \alpha_{i,j})$	Pairwise transmission likelihood of edge (i, j)
$F(t_j\|t_i, \alpha_{i,j})$	Cumulative density function of edge (i, j)
$S(t_j\|t_i; \alpha_{i,j})$	*Survival function* of edge (i, j)
$H(t_j\|t_i; \alpha_{i,j})$	*Hazard function*, or instantaneous activation rate, of edge (i, j)
$g(\mathbf{A})$	Prior likelihood on the transmission rates $\mathbf{A}$
$\mathcal{A}$	Support of the prior likelihood on the transmission rates $g(\mathbf{A})$

Table 5.1: Table of symbols for Chapter 5.

5.2.1 Data

We observe multiple waves of contagions that propagate on a fixed population of N nodes. As the contagion spreads from activated to non-activated nodes it creates a *cascade*. For each contagion c, we observe a cascade $\mathbf{t}^c$, which is simply a record of observed node activation times. In an information propagation setting, each cascade corresponds to a different piece of information and the activation time of a node is simply the time when the node first heard of or mentioned the piece of information.

We record a set $\mathcal{C}$ of cascades $\{\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}\}$. A cascade $\mathbf{t}^c = (t_1^c, \ldots, t_N^c)$ is an N-dimensional vector recording when each of N nodes got activated by the contagion c during a time interval of finite length T^c. Thus, $t_k^c \in [t_0, t_0 + T^c] \cup \{\infty\}$, where t_0 is the activation time of the first node. Symbol ∞ labels nodes that are not activated by the contagion c during observation window $[t_0, t_0 + T^c]$ – it does not imply that nodes are never activated. Lengthening the observation window T^c increases the number of observed activations within a cascade c and results in a more representative sample of the underlying dynamics. However, these advantages must be weighed against the cost of observing for longer periods. For simplicity we assume $T^c = T$ for all cascades; the results generalize trivially. We assume contagions spread at different rates $\alpha_{i,j}$ across different edges of the underlying unobserved network $\mathcal{G}$. Thus, we consider the network to be weighted. Contagions often propagate simultaneously (Myers and Leskovec, 2012; Prakash et al., 2012b) over the same network but we assume each contagion to propagate independently of each other. We illustrate this process in Figure 5.1.

Given a set of node activation times of many different contagions, our goal is to infer the transmission rates $\mathbf{A} = [\alpha_{j,i}]$ and underlying network over which contagions propagated. Importantly, the time-stamps assigned to nodes in each cascade induce the structure of a directed acyclic graph (DAG) on the network (which is *not* acyclic in general). Thus, it is meaningful to refer to parents and children within a cascade, but not on the network. The DAG structure dramatically simplifies the computational complexity of the inference problem.

5.2.2 Pairwise interactions

As previously for unweighted networks (Section 4.2), we describe the pairwise interactions between nodes using three concepts: pairwise transmission rates $\alpha_{i,j}$, prior probabilities of transmission $\beta_{i,j}$, and pairwise transmission likelihoods $f(t_i|t_j, \alpha_{i,j})$. The transmission rate $\alpha_{i,j}$ of an edge $(i, j) \in \mathcal{E}$ quantifies how frequently any contagion spreads from node i to node j or, in other words, the *latency* of the edge (i, j). The prior transmission probability $\beta_{i,j}$ of an edge (i, j) quantifies the probability that a contagion would eventually spread from node i to node j for arbitrarily large t_j. Finally, the pairwise transmission likelihood $f(t_j|t_i; \alpha_{i,j})$ of an edge $(i, j) \in \mathcal{E}$ is the conditional likelihood of transmission from node i, activated at time t_i, to node j. We refer the reader to Section 2.3 for an in-depth discussion of all three concepts. Now, we highlight the key assumptions on the pairwise interactions of our model of diffusion over weighted networks:

First, since we assume networks to be weighted, we account for the general case of heterogeneous pairwise transmission rates, *i.e.*, activations can occur at different transmission rates over different edges of a network.

Second, for any contagion c, we observe activations up to a finite time horizon $T^c \to \infty$. This contrasts with our approach to diffusion over unweighted networks in Section 4, where we consider arbitrarily large time horizons $T^c \to \infty$. Then, given a node i, activated at t_i, and a transmission rate $\alpha_{i,j} > 0$, the probability of survival of node j up to the time horizon T^c will be always greater than 0, $S(t_j|t_i;\alpha_{i,j}) = \int_{T^c}^{\infty} f(t_j|t_i;\alpha_{i,j}) > 0$. Then, we can assume, for simplicity, the prior probability of transmission $\beta_{i,j}$ to be 1 and yet not always observe node j to get infected before T^c.

Third, the transmission likelihood for every edge (i,j) depends on the activation times (t_i, t_j) and a transmission rate $\alpha_{i,j}$. For simplicity, we consider well-known parametric models used previously in the literature (refer to Table 2.1). Importantly, as $\alpha_{j,i} \to 0$, the likelihood of transmission tends to zero and the expected transmission time becomes arbitrarily long and always larger than the time horizon T^c. We will later allow transmission rates $\alpha_{j,i}$ to change over time. In particular, we will allow the transmission rates $\alpha_{j,i}$ to change across cascades but not within a cascade. Allowing edge transmission rates to dynamically increase and decay over time will enable us to infer dynamic, time-varying, diffusion networks.

We recall some additional standard notation, defined in Section 2.3: the cumulative density function, denoted $F(t_j|t_i;\alpha_{i,j})$, the *survival function*, $S(t_j|t_i;\alpha_{i,j}) = 1 - F(t_j|t_i;\alpha_{j,i})$, and the *hazard function*, or instantaneous activation rate, $H(t_i|t_j;\alpha_{j,i}) = -S'(t_i|t_j;\alpha_{j,i})/S(t_i|t_j;\alpha_{j,i}) = f(t_i|t_j;\alpha_{j,i})/S(t_i|t_j;\alpha_{j,i})$. Importantly, the log-survival and hazard functions of the parametric models we consider throughout this dissertation are simple (see Table 2.1). We refer the reader to Lawless (1982) for a more extensive discussion of survival and hazard functions.

5.2.3 Probability of survival given a cascade

We compute the probability that a node survives unactivated until time t_i, given that some of its parents are already activated. Consider a cascade $\mathbf{t} := (t_1, \ldots, t_N)$.

Since each activated node k may activate i independently, the probability that nodes $1 \ldots N$ do *not* activate node i by time t_i is the product of the survival functions of the activated nodes $1 \ldots N | t_k \leq t_i$ targeting i,

$$S(t_i | t_1, \ldots, t_N \setminus t_i; \mathbf{A}) = \prod_{t_k \leq t_i} S(t_i | t_k; \alpha_{k,i}), \tag{5.1}$$

where $\mathbf{A} := \{\alpha_{i,j} \,|\, i, j = 1, \ldots, n, i \neq j\}$.

5.2.4 Likelihood of a cascade

Consider a cascade $\mathbf{t} := (t_1, \ldots, t_N)$. We first compute the likelihood of the observed activations $\mathbf{t}^{\leq T} = (t_1, \ldots, t_N | t_i \leq T)$. Since we assume activations are conditionally independent given the parents of the activated nodes, the likelihood factorizes over nodes as

$$f(\mathbf{t}^{\leq T}; \mathbf{A}) = \prod_{t_i \leq T} f(t_i | t_1, \ldots, t_N \setminus t_i; \mathbf{A}). \tag{5.2}$$

Computing the likelihood of a cascade thus reduces to computing the conditional likelihood of activating each node given the rest of the cascade. As in the independent cascade model (Kempe et al., 2003), we assume that a node gets activated once the *first* parent activates the node. Given an activated node i, we compute the probability of a potential parent j to be the first parent by applying Eq. 5.1,

$$f(t_i | t_j; \alpha_{j,i}) \times \prod_{j \neq k, t_k < t_i} S(t_i | t_k; \alpha_{k,i}). \tag{5.3}$$

We now compute the conditional likelihoods of Eq. 5.2 by summing over the likelihoods of the mutually disjoint events that each potential parent is the first parent,

$$f(t_i | t_1, \ldots, t_N \setminus t_i; \mathbf{A}) = \sum_{j : t_j < t_i} f(t_i | t_j; \alpha_{j,i}) \times \prod_{j \neq k, t_k < t_i} S(t_i | t_k; \alpha_{k,i}). \tag{5.4}$$

By Eq. 5.2 the likelihood of the activations in a cascade is

$$f(\mathbf{t}^{\leq T}; \mathbf{A}) = \prod_{t_i \leq T} \sum_{j:t_j < t_i} f(t_i|t_j; \alpha_{j,i}) \times \prod_{k:t_k < t_i, k \neq j} S(t_i|t_k; \alpha_{k,i}). \tag{5.5}$$

Removing the condition $k \neq j$ makes the product independent of j,

$$f(\mathbf{t}^{\leq T}; \mathbf{A}) = \prod_{t_i \leq T} \prod_{k:t_k < t_i} S(t_i|t_k; \alpha_{k,i}) \times \sum_{j:t_j < t_i} \frac{f(t_i|t_j; \alpha_{j,i})}{S(t_i|t_j; \alpha_{j,i})}, \tag{5.6}$$

and we can replace the ratios in Eq. 5.6 with hazard functions:

$$f(\mathbf{t}^{\leq T}; \mathbf{A}) = \prod_{t_i \leq T} \prod_{k:t_k < t_i} S(t_i|t_k; \alpha_{k,i}) \times \sum_{j:t_j < t_i} H(t_i|t_j; \alpha_{j,i}). \tag{5.7}$$

Now, we note that Eq. 5.7 only considers activated nodes. However, the fact that some nodes are *not* activated during the observation window is also informative. We therefore add the multiplicative survival term from Eq. 5.1:

$$f(\mathbf{t}; \mathbf{A}) = \prod_{t_i \leq T} \prod_{t_m > T} S(T|t_i; \alpha_{i,m}) \times \prod_{k:t_k < t_i} S(t_i|t_k; \alpha_{k,i}) \sum_{j:t_j < t_i} H(t_i|t_j; \alpha_{j,i}). \tag{5.8}$$

Assuming independent cascades, the likelihood of a set of cascades $\mathcal{C} = \{\mathbf{t}^1, \ldots, \mathbf{t}^{|\mathcal{C}|}\}$ is the product of the likelihoods of the individual cascades given by Eq. 5.8:

$$f(\{\mathbf{t}^1, \ldots, \mathbf{t}^{|\mathcal{C}|}\}; \mathbf{A}) = \prod_{\mathbf{t}^c \in \mathcal{C}} f(\mathbf{t}^c; \mathbf{A}). \tag{5.9}$$

5.2.5 Relation to unweighted networks

In this section, we show how the likelihood of a cascade $\mathbf{t}^c$ under our diffusion model for weighted networks, given by Eq. 5.8, relates to likelihood under our diffusion model for unweighted networks, given by Eq. 4.3, by considering arbitrarily large time horizon T and equal transmission rates $\alpha_{i,j} = \alpha$.

Consider equal transmission rates $\alpha_{i,j} = \alpha$ for every edge $(i, j) \in \mathcal{E}$ and an arbitrarily large time horizon T, as in Section 4.2. Then, by definition of β, $\lim_{T \to \infty} S(T|t_i; \alpha) =$

$(1-\beta)$ if $(i,j) \in \mathcal{E}$. Now, we can rewrite Eq. 5.8 as:

$$f(\mathbf{t}|\mathcal{G}) = \prod_{t_i \leq T} \sum_{(j,i)\in\mathcal{E}\,:\,t_j<t_i} f(t_i|t_j;\alpha) \prod_{(k,i)\in\mathcal{E}\,:\,k\neq j, t_k<t_i} S(t_i|t_k;\alpha) \times \prod_{t_n\leq T} \prod_{(n,m)\in\mathcal{E}\,:\,t_m>T} (1-\beta). \tag{5.10}$$

By definition, $f(t_i|t_j;\alpha_{j,i}) = 0$ and $S(t_i|t_j;\alpha_{j,i}) = 1$ for either $(j,i) \notin \mathcal{E}$ or $(j,i) \in \mathcal{E}$ and $t_j > t_i$. Then, we can rewrite Eq. 5.10 as:

$$\begin{aligned} f(\mathbf{t}|\mathcal{G}) &= \sum_{j\in\mathcal{V}} \prod_{t_i\leq T} \left(f(t_i|t_j;\alpha) \prod_{(k,i)\in\mathcal{E}:k\neq j} S(t_i|t_k;\alpha) \right) \times \prod_{t_n\leq T} \prod_{(n,m)\in\mathcal{E}\,:\,t_m>T} (1-\beta) \\ &= \sum_{\mathcal{T}\in\mathcal{T}_c(\mathcal{G})} \prod_{(j,i)\in\mathcal{E}_T} \left(f(t_i|t_j;\alpha) \prod_{(k,i)\in\mathcal{E}\setminus\mathcal{E}_T} S(t_i|t_k;\alpha) \right) \times (1-\beta)^r, \end{aligned}$$

where r counts the number of edges that did not activate and failed to transmit the contagion: $r = \sum_{u\in V_T} d_{out}(u) - q$, and $d_{out}(u)$ is the out-degree of node u in graph $\mathcal{G}$. However, this equation still differs from the likelihood of a cascade $\mathbf{t}^c$ under our diffusion model for unweighted networks, given by Eq. 4.3. In order to recover it, we need to further assume that the prior transmission probability β of an edge (j,i) matches the probability of survival of node i to the rest of potential parents k: $\prod_{(k,i)\in\mathcal{E}\setminus\mathcal{E}_T} S(t_i|t_k;\alpha) \approx \beta$. Then,

$$\begin{aligned} f(\mathbf{t}|\mathcal{G}) &\approx \sum_{\mathcal{T}\in\mathcal{T}_c(\mathcal{G})} \prod_{(j,i)\in\mathcal{E}_T} f(t_i|t_j;\alpha) \times \beta^q (1-\beta)^r \\ &= \sum_{\mathcal{T}\in\mathcal{T}_c(\mathcal{G})} f(\mathbf{t}^c|\mathcal{T}) P(\mathcal{T}|\mathcal{G}), \end{aligned}$$

where $q = |\mathcal{E}_T| = |\mathcal{V}_T| - 1$ is the number of edges in $\mathcal{T}$ and counts the edges over which the diffusion process successfully propagated.

5.2.6 Three weighted network inference problems

Given a static weighted network with constant transmission rates $\alpha_{j,i}$, the network inference problem reduces to solving a maximum likelihood problem:

Problem 2 (Static Network Inference)**.** *Given an observed set of cascades* $\mathcal{C} = \{\mathbf{t}^1, \ldots, \mathbf{t}^{|\mathcal{C}|}\}$*, our goal is to find the underlying transmission rates* $\alpha_{j,i}$ *by solving the following maximum likelihood (ML) optimization problem:*

$$\begin{array}{ll} \textit{minimize}_{\mathbf{A}} & -\sum_{c \in \mathcal{C}} \log f(\mathbf{t}^c; \mathbf{A}) \\ \textit{subject to} & \alpha_{j,i} \geq 0,\, i,j = 1, \ldots, N, i \neq j, \end{array} \tag{5.11}$$

where $\mathbf{A} := \{\alpha_{j,i} \,|\, i,j = 1, \ldots, n, i \neq j\}$ *are the variables. The edges of the network are those pairs of nodes with transmission rates* $\alpha_{j,i} > 0$*.*

Now, we generalize the network inference problem to dynamic networks with transmission rates $\alpha_{j,i}(t)$ that may change over time.

Problem 3 (Dynamic Network Inference)**.** *Given a time* t *and a set of recorded cascades by time* t*,* $\mathcal{C}_t = \{\mathbf{t}^1, \ldots, \mathbf{t}^{|\mathcal{C}_t|}\}$*, our goal is to find the optimal transmission rates* $\alpha_{j,i}(t)$ *by solving the following maximum likelihood (ML) optimization problem:*

$$\begin{array}{ll} \textit{minimize}_{\mathbf{A}(t)} & -\sum_{c \in \mathcal{C}_t} w_c(t) \log f(\mathbf{t}^c; \mathbf{A}(t)) \\ \textit{subject to} & \alpha_{j,i}(t) \geq 0,\, i,j = 1, \ldots, N, i \neq j, \end{array} \tag{5.12}$$

where $w_c(t) \geq 0$ *is a weight that penalizes how old cascade* c *is at time* t *and* $\mathbf{A}(t) := \{\alpha_{j,i}(t) \,|\, i,j = 1, \ldots, n, i \neq j\}$ *are the variables. The intuition here is that diffusion network smoothly changes over time and that recent cascades have higher importance in determining current network structure than old cascades. Thus at any point in time we can solve the above optimization problem to obtain the structure of the diffusion network at that particular time.*

The dynamic network inference problem defined by Eq. 5.12 reduces to the static network inference problem defined by Eq. 5.11 when we set all weights $w_c(t)$ to be equal and constant over time.

Finally, in some scenarios we may have access to additional information that lets us estimate a prior likelihood on the transmission rates $\alpha_{j,i}(t)$. For example, in information networks, a blog may sometimes link its sources, and therefore we can compute a prior on the transmission rates from the sources to the blog using those links. In such cases, we can solve instead a maximum a posteriori optimization problem.

Problem 4 (Network Inference with Prior Likelihood)**.** *Given a time t and a set of recorded cascades by time t, $\mathcal{C}_t = \{\mathbf{t}^1, \ldots, \mathbf{t}^{|\mathcal{C}_t|}\}$, and a prior likelihood $g(\mathbf{A}(t))$ on the transmission rates $\alpha_{j,i}(t)$, our goal is to find the optimal transmission rates $\alpha_{j,i}(t)$ by solving the following maximum a posteriori (MAP) optimization problem:*

$$\begin{array}{ll} \textit{minimize}_{\mathbf{A}(t)} & -\sum_{c\in\mathcal{C}_t} w_c(t) \log f(\mathbf{t}^c; \mathbf{A}(t)) - \log g(\mathbf{A}(t)) \\ \textit{subject to} & \mathbf{A}(t) \in \mathcal{A}, \\ & \alpha_{j,i}(t) \geq 0,\, i,j = 1,\ldots,N, i \neq j, \end{array} \tag{5.13}$$

where $w_c(t) \geq 0$ is a weight that penalize how old cascade c is at time t, $\mathbf{A}(t) := \{\alpha_{j,i}(t)\,|\, i,j = 1,\ldots,n, i \neq j\}$ are the variables and $\mathcal{A}$ is the support of the prior likelihood $g(\cdot)$.

5.3 NetRate

5.3.1 Algorithm

Perhaps surprisingly, the solutions to the static and the dynamic network inference problems defined by Eqs. 5.11 and 5.12 are unique, computable and consistent:

Theorem 7. *Given log-concave survival functions and concave hazard functions in the parameter(s) of the pairwise transmission likelihoods, the static and dynamic network inference problem defined by Eqs. 5.11 and 5.12 are convex in $\mathbf{A}$.*

Proof. By Eq. 5.8, the log-likelihood of a cascade is

$$L(\mathbf{t}^c; \mathbf{A}) = \Psi_1(\mathbf{t}^c; \mathbf{A}) + \Psi_2(\mathbf{t}^c; \mathbf{A}) + \Psi_3(\mathbf{t}^c; \mathbf{A}), \tag{5.14}$$

where,

$$\begin{aligned}
\Psi_1(\mathbf{t}^c; \mathbf{A}) &= \sum_{i:t_i \leq T} \sum_{t_m > T} \log S(T|t_i; \alpha_{i,m}), \\
\Psi_2(\mathbf{t}^c; \mathbf{A}) &= \sum_{i:t_i \leq T} \sum_{j:t_j < t_i} \log S(t_i|t_j; \alpha_{j,i}), \\
\Psi_3(\mathbf{t}^c; \mathbf{A}) &= \sum_{i:t_i \leq T} \log \left(\sum_{j:t_j < t_i} H(t_i|t_j; \alpha_{j,i}) \right).
\end{aligned}$$

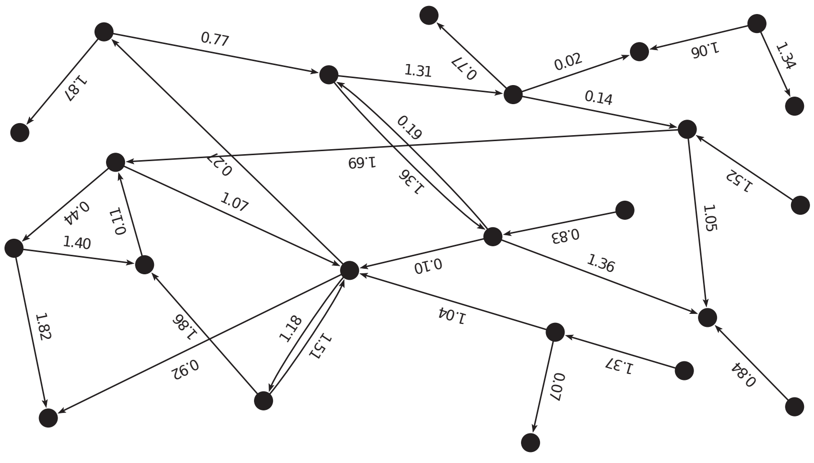
If all pairwise transmission likelihoods between pairs of nodes in the network have log-concave survival functions and concave hazard functions in the parameter(s) of the pairwise transmission likelihoods, then convexity of Eqs. 5.11 and 5.12 follows from linearity, composition rules for concavity, and concavity of the logarithm. □

Corollary 8. *The static and dynamic network inference problem defined by Eqs. 5.11 and 5.12 are convex for the exponential, power-law and Rayleigh models.*

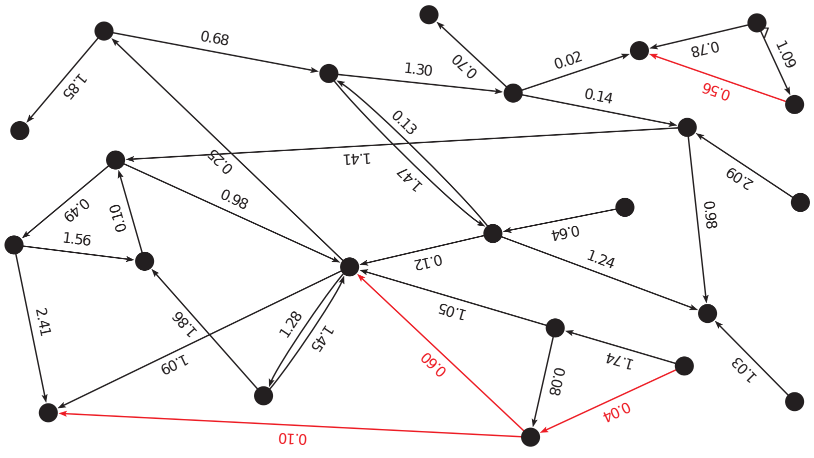
Theorem 9. *The maximum likelihood estimator $\hat{\alpha}$ given by the solution of Eq. 5.11 is consistent.*

Proof Sketch. We check the criteria for consistency of identification, continuity and compactness (Newey and McFadden, 1994). The log-likelihood in Eq. 5.14 is a continuous function of $\mathbf{A}$ for any fixed set of cascades $\{\mathbf{t}^1 \ldots \mathbf{t}^{|\mathcal{C}|}\}$, and each α defines a unique function $\log f(\cdot|\mathbf{A})$ on the set of cascades. Finally, note that $L \to -\infty$ for both $\alpha_{ij} \to 0$ and $\alpha_{ij} \to \infty$ for all i, j so we lose nothing imposing upper and lower bounds thus restricting to a compact subset.

Similarly, the solution to the maximum a posteriori optimization problem defined by Eq. 5.13 is also unique, computable and consistent if the prior likelihood on $\mathbf{A}$ is log-concave. In the remainder of the chapter, we focus on the maximum likelihood approach for both static and dynamic networks, and we call our network inference method NetRate.



(a) True network G



(b) Inferred network $\hat{\mathcal{G}}$

Figure 5.2: Accuracy of NetRate in a small core-periphery Kronecker network. Panel (a) shows the true network G and panel (b) shows the inferred network by NetRate from 200 cascades. Red edges denote mistakes and the number over each edge denotes the (inferred) pairwise transmission rate. NetRate recovers all the true edges and outputs only four false edges.

5.3.2 Properties of NetRate

We highlight some common features of the solutions to the network inference problem for the exponential, power-law and Rayleigh models. First, to illuminate the discussion, we revisit the terms constituting the log-likelihood Eq. 5.14 for the four transmission models in Table 2.1.

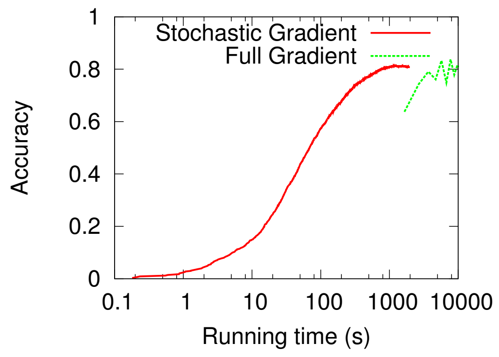
The Ψ_1 and Ψ_2 terms contribute a positively weighted l_1-norm on vector $\mathbf{A}$ that encourages sparse solutions (Boyd and Vandenberghe, 2004). The penalty arises naturally within the probabilistic model so that heuristic penalty terms to encourage sparsity are not necessary. Each term of the l_1-norm is positively weighted by a linear function (exponential model), a logarithm (power-law), a quadratic function (Rayleigh) or a monomial of degree k of the activation times. Sparse solutions are desirable since real networks are usually sparse, as argued for unweighted networks in Section 4.

The Ψ_2 term penalizes edges $k \to i$ based on the activation time difference $t_i - t_k$. Edges transmitting activations slowly are heavily penalized and conversely. The Ψ_1 term penalizes edges $i \to j$ targeting *unactivated* nodes j based on the time $T - t_i$ till the observation window cutoff. Lengthening the observation window produces harsher penalties – however, it also allows further activations. The penalties are finite, *i.e.*, if no activation of node j is observed, we can only say that it has survived until time T. There is insufficient evidence to claim j will never be activated since our data is *right-censored* (Aalen et al., 2008). NetRate does not use empirically ungrounded parameters (such as number of edges k and penalty factor ρ used by NetInf and ConNIe respectively) to leap from not observing an activation to inferring it is impossible. Instead, NetRate infers that the most likely explanation of the observed data does not require transmission across certain edges.

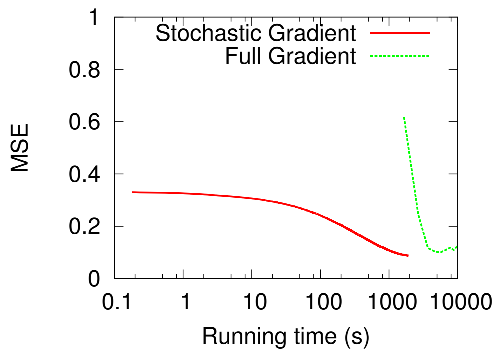
The Ψ_3 term ensures activated nodes have at least one parent since otherwise the objective function would be negatively unbounded, *i.e.*, $\log 0 = -\infty$. Moreover, our formulation encourages a natural diminishing property on the number of parents of a node – since the logarithm grows *slowly*, it weakly rewards activated nodes for having many parents. We have found a similar diminishing property on the number of parents, submodularity, on our approach to unweighted networks in Section 4.

5.3.3 Solving NetRate

Initially, we solved both the static and dynamic network inference problem using `CVX`, a general purpose package for specifying and solving convex programs (Grant



(a) Accuracy



(b) MSE

Figure 5.3: Accuracy and mean square error (MSE) against running time for a 1,024 node, 3,161 edge static core-periphery Kronecker network with exponential model for 10,000 cascades. Longer running times correspond to more iterations. A stochastic gradient implementation of NetRate is approximately one order of magnitude faster than a full gradient implementation.

and Boyd, 2010), and we publicly released an open source implementation[1]. Then, in order to increase scalability, we developed a stochastic gradient (SG) descent implementation of our method, which we called InfoPath, and we also publicly released an open source implementation[2]. Figure 5.3 illustrates how our stochastic gradient implementation of NetRate (a.k.a. InfoPath) is approximately one order of magnitude faster than a full gradient implementation. For the sake of fairness, since InfoPath was coded in C++, we compared with a full gradient (non-stochastic) descent implementation of NetRate in C++ instead of the Matlab code which uses CVX, which was slower.

Stochastic gradient (SG) descent

Stochastic gradient (SG) descent methods have been shown to be extremely successful for taking advantage of the structure exhibited by the optimization problems stated in Eqs. 5.11 and 5.12. They have received increasing attention in the machine

[1]A Matlab implementation of NetRate using CVX is available in a supporting website (NetRate, 2011)

[2]A C++ stochastic gradient (SG) descent implementation of NetRate, which we called InfoPath, is available in a supporting website (InfoPath, 2013)

learning literature (Agarwal and Duchi, 2011; Bach et al., 2011; Blatt et al., 2008; Duchi et al., 2011). Although many convex optimization methods based on stochastic gradient descent have been proposed, we have found that in practice the basic projected stochastic gradient method (Robbins and Monro, 1951) works well enough for our problem. Other more sophisticated methods, like the stochastic average gradient (Roux et al., 2012) or incremental average gradient (Blatt et al., 2008) do not offer a significant advantage. Therefore, we proceed with the basic stochastic gradient method in the remainder of the chapter.

In the static network inference problem defined by Eq. 5.11, the projected SG method (Robbins and Monro, 1951) uses iterations of the form:

$$\alpha_{j,i}^{k} = \left(\alpha_{j,i}^{k-1} - \gamma_k \nabla_{\alpha_{j,i}} L_{c_k}(\mathbf{A}^{k-1})\right)^{+}, \tag{5.15}$$

where $\nabla_{\alpha_{j,i}} L_{c_k}(\cdot)$ is the gradient of the log-likelihood $L_c(\cdot)$ with respect to the transmission rate $\alpha_{j,i}$, γ_k is a step-size, $(z)^{+} = \max(0, z)$, and cascade c_k is sampled (with replacement) uniformly at random from $\mathcal{C}$. The gradients for all four edge transmission models are given in Table 5.2.

In the dynamic network inference problem defined by Eq. 5.12, the projected SG method (Robbins and Monro, 1951) uses iterations of the form:

$$\alpha_{j,i}^{k}(t) = \left(\alpha_{j,i}^{k-1}(t) - \gamma_k \nabla_{\alpha_{j,i}} L_{c_k}(\mathbf{A}^{k-1}(t))\right)^{+}, \tag{5.16}$$

where $\nabla_{\alpha_{j,i}} L_{c_k}(\cdot)$ is the gradient of the log-likelihood $L_c(\cdot)$ with respect to the transmission rate $\alpha_{j,i}$, γ_k is a step-size, $(z)^{+} = \max(0, z)$, and cascade c_k is sampled (with replacement, not uniformly) from $\mathcal{C}_t$. In this case, instead using all historic data and then explicitly penalizing each cascade by a different weighting factor $w_c(t)$, we use a different, more scalable approach. We sample cascades with replacement where the probability of a cascade being sampled decays with the age of the cascade. This way recent cascades get sampled more often and thus implicitly hold higher importance when inferring the network. In practice, we achieve a significant speed up using this approach. Moreover, in our dynamic network inference problem, the transmission rates usually vary smoothly. This means that stochastic gradient descent is a natural

Model	**Cascade gradient for unactivated**	**Cascade gradient for activated**
	$\nabla_{\alpha_{j,i}} L_c(\mathbf{A})$	$\nabla_{\alpha_{j,i}} L_c(\mathbf{A})$
Exponential	$T - t_j^c$	$(t_i^c - t_j^c) - \frac{1}{\sum_{l:t_l^c<t_i^c} \alpha_{l,i}}$
Power-law	$\log\left(\frac{T-t_j^c}{\delta}\right)$	$\log\left(\frac{t_i^c-t_j^c}{\delta}\right) - \frac{(t_i^c-t_j^c)^{-1}}{\sum_{l:t_l^c<t_i^c} \alpha_{l,i}(t_i^c-t_l^c)^{-1}}$
Rayleigh	$\frac{(T-t_j^c)^2}{2}$	$\frac{(t_i^c-t_j^c)^2}{2} - \frac{t_i^c-t_j^c}{\sum_{l:t_l^c<t_i^c} \alpha_{l,i}(t_i^c-t_l^c)}$
Weibull	$(T - t_j^c)^k$	$(t_i^c - t_j^c)^k - \frac{k(t_i^c-t_j^c)^{k-1}}{\sum_{l:t_l^c<t_i^c} k\alpha_{l,i}(t_i^c-t_l^c)^{k-1}}$

Table 5.2: Cascade gradients for transmission models.

method since we can use the inferred network from the previous time step as initialization for the inference procedure in the current time step. We find that setting the starting point $\alpha_{j,i}^0$ of each transmission rate $\alpha_{j,i}$ to the last outputted estimate of the transmission rate allow us to further speed up the algorithm.

Importantly, in each iteration k of the projected stochastic gradient method for both static and dynamic networks, we only need to compute the gradients $\nabla_{\alpha_{j,i}} L_{c_k}(\mathbf{A}^k)$ for edges (j, i) such that node j has been activated in cascade c_k, and the iteration cost and convergence rate are independent of $|\mathcal{C}|$ (Bach et al., 2011; Nemirovski et al., 2009). Rigorous theoretical analysis of convergence turns out to be a challenging problem which we leave for future work. However, we would like to point out that such analysis typically assumes the gradients $\nabla_{\mathbf{A}} L_c(\mathbf{A}^k)$ to be either bounded above by a constant M, $||\nabla_{\mathbf{A}} L_c(\mathbf{A})|| \leq M$, or Lipschitz-continuous with constant L, $||\nabla_{\mathbf{A}} L_c(\mathbf{A}_2) - \nabla_{\mathbf{A}} L_c(\mathbf{A}_1)|| \leq L||\mathbf{A}_2 - \mathbf{A}_1||$. In our problem, these conditions are violated if at any iteration k, there is a node i activated in cascade c_k such that $H(t_i^{c_k}|t_j^{c_k}; \alpha_{j,i}^{k-1}) = 0$ $\forall j : t_j^{c_k} < t_i^{c_k}$, *i.e.*, node i has no parents that *explain* the activation at $t_i^{c_k}$, and the objective function is positively unbounded. In practice, we avoid this scenario by introducing a lower bound on *feasible* transmission rates, so that $\alpha_{j,i} \geq \varepsilon$. A transmission rate $\alpha_{j,i}$ is *feasible* if there is at least one cascade in which both node j and i get activated. When outputting a solution, we simply omit transmission rates with value ε.

Algorithm 4 Stochastic gradient implementation of NetRate for static networks

```
Require: $C, K$
  while $k < K$ do
    $c_k \leftarrow$ uniform-sampling($C$);
    for all $(j, i) : t_j^{c_k} < t_i^{c_k}$ do
      $\alpha_{j,i}^k = \left(\alpha_{j,i}^{k-1} - \gamma_k \nabla_{\alpha_{j,i}} L_{c_k}(\mathbf{A}^{k-1})\right)^+$;
    end for
    k = k+1;
  end while
  $\mathbf{A}^* \leftarrow \mathbf{A}^{K-1}$;
  return $\mathbf{A}^*$;
```

Aging edges in dynamic networks

Our algorithm automatically penalizes edges (j, i) when the source node j gets activated and the target node i does not. In other words, in each iteration k of the (stochastic) gradient descent method, we update transmission rates $\alpha_{j,i}^k$ if node j gets activated in cascade c_k. Therefore, an edge (j, i) gets penalized if node j gets activated in at least one cascade c_k. In the dynamic setting, we introduce the additional assumption that *unused* edges decay exponentially. In online media, for example, bloggers typically pay less attention to news sites or blogs that have not been activated recently. If a node j has not been activated recently, we would like the *unused* edges (j, i) to decay and eventually vanished, or equivalently the transmission rates $\alpha_{j,i}$ to converge to zero. We incorporate this observation by multiplying the transmission rates of unused edges by an *aging* factor ρ every time t we solve the dynamic network inference problem. Our implementation penalizes edges (j, i) where node j never gets activated. We use an aging factor $\rho = 0.95$ in our experiments.

Cascade sampling in dynamic networks

In Eq. 5.16, instead of sampling cascades uniformly at random and explicitly penalizing each cascade by a different weighting factor $w_c(t)$, we achieve a significant speed up by sampling cascades using a procedure that penalizes old cascades and sets $w_c(t) = 1$ for all cascades. There are many different sampling procedures. For

Algorithm 5 Stochastic gradient implementation of NetRate for dynamic networks

```
Require: $C_t, K, T, \rho$
  while $k < K$ do
    $c_k \leftarrow$ cascade-sampling($C_t, T$);
    for all $(j, i) : t_j^{c_k} < t_i^{c_k}$ do
      $\alpha_{j,i}^k = \left(\alpha_{j,i}^{k-1} - \gamma_k \nabla_{\alpha_{j,i}} L_{c_k}(\mathbf{A}^{k-1})\right)^+$;
    end for
    for all $(j, i) : \alpha_{j,i}^{k-1} > 0, t_j^{c_k} \to \infty$ do
      $\alpha_{j,i}^k = \rho \alpha_{j,i}^{k-1}$;
    end for
    k = k+1;
  end while
  $\mathbf{A}^* \leftarrow \mathbf{A}^{K-1}$;
  return $\mathbf{A}^*$;
```

simplicity, we use windowed uniform or windowed exponential sampling. Windowed means that when solving the network inference problem for time t, we only sample cascades that started in the time window $(t - T_s, T_s)$. Here, we encounter an important tradeoff. The shorter the sampling time window T_s in the stochastic gradient descent, the quicker our algorithm tracks changes in transmission rates. However, a short sampling time window results in less reliable estimates because we sample fewer cascades. To track changes quickly, we therefore need to observe many cascades over time.

Distributed optimization

The optimization problem splits into N subproblems, one for each node i, in which we find $N - 1$ rates $\alpha_{j,i}$, $j = 1, \ldots, N \setminus i$. The computation can be performed in parallel, obtaining local solutions that are globally optimal. Importantly, each node's computation only requires the activation times of other nodes in cascades it belongs to. This may allow us to scale NetRate beyond hundred of thousands of nodes.

Unfeasible rates

If a pair (j,i) is not in any common cascades, $\alpha_{j,i}$ only arises in the non-positive term Ψ_3 in Eq. 5.14, so the optimal $\alpha_{j,i}$ is zero. We therefore simply modify the optimization problem by setting $\alpha_{j,i}$ to zero – we remove $\alpha_{j,i}$ from the optimization problem. In a network with hundreds of thousands of nodes (and in principle billion of edges), this tweak can speed up inference by several orders of magnitude.

5.3.4 Experiments on synthetic data

In this section, we validate NetRate by evaluating its performance on synthetic static and dynamic weighted networks. First, we compare the performance of NetRate in static weighted networks against one of our network inference algorithms for unweighted networks, NetInf, described in Section 4.3.1, and one state of the art network inference algorithm for weighted networks, ConNIe Myers and Leskovec (2010). Second, we analyze the performance of our algorithm in static networks as a function of cascade coverage, time horizon, transmission rates distributions, exogenous factors, and noise. Finally, we analyze its performance in dynamic networks as a function of the transmission rate temporal trend and the sampling window when using the stochastic gradient descent implementation, InfoPath.

In general, in all our experiments we proceed as follows: We are given a true diffusion static (dynamic) network $\mathcal{G}^*$ ($\mathcal{G}^*(t)$) and fixed (variable) transmission rates $\mathbf{A}$ ($\mathbf{A}(t)$), and then we simulate the propagation of a set of contagions c over the network. Diffusion of each contagion creates a cascade and for each cascade, we record the node activation times t_i. Root nodes of cascades are chosen at random. Once a node is activated, the transmission likelihoods of outgoing edges determine the activation times of its neighbors. We record the time of the first activation if a node is activated more than once. Activations are not observed after a pre-specified time horizon T. Then, given these activation times (*i.e.*, set of cascades), we aim to recover the static (dynamic) network using NetRate. For example, Figure 5.2(a) shows a small static diffusion network G of 23 nodes and 30 directed edges. Using the exponential model we generated 200 cascades. Now given the cascades, NetRate returns the network $\hat{\mathcal{G}}$ in Figure 5.2(b). Our method recovered G almost perfectly by making only four errors (red edges), and it outputs pairwise transmission rates (numbers over edges) that are very close to the true values.

Experimental setup

Our experimental methodology for static networks with fixed transmission rates over time is composed of the following steps:

1. Ground truth network $\mathcal{G}^*$.
2. Cascade generation: transmission time model with parameters $\alpha_{i,j}$ and time horizon T.

(1) Ground truth network $\mathcal{G}^*$: As in Section 4.3.2, we consider two models of directed real-world networks to generate $\mathcal{G}^*$, namely, the Forest Fire model (Leskovec et al., 2005) and the Kronecker Graphs model Leskovec et al. (2010). For Kronecker networks, we consider three sets of parameters that produce networks with a very different global network structure: a random network (Erdős and Rényi, 1960), a core-periphery network (Leskovec et al., 2008) and a network with hierarchical community structure (Clauset et al., 2008). A brief introduction to Forest Fire and Kronecker Graph models can be found in Section 2.2.2.
(2) Cascade generation: For cascade generation, we need both a ground truth network structure and transmission rates for each edge (i, j). Therefore, we draw transmission rates each edge (i, j) from a uniform distribution, a Gaussian distribution or a Rayleigh distribution. We control the transmission rate variance across edges in the network by tuning the parameters values of the distributions. The transmission rate for an edge (i, j) models how fast the information spreads from node i to node j in social networks. If not specified, $\alpha \sim U(0.01, 1)$ for the exponential and Rayleigh models and $\alpha \sim U(0.01, 2)$ for the power law. Then, we simulate cascades on G^* using the generative model defined in Section 4.2, and we record activations up to a pre-specified time horizon T. Then, given these activation times (*i.e.*, set of cascades), we aim to recover the static network $\mathcal{G}^*$ using NetRate.

Our experimental methodology for dynamic networks with variable transmission rates over time is composed of the following steps:

1. Ground truth network $\mathcal{G}^*(t)$.
2. Cascade generation: transmission time model with parameters $\alpha_{i,j}(t)$ and time horizon T.
3. Length of the sampling time window.

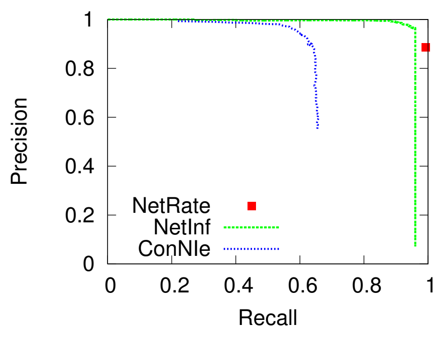
(1) Ground truth network $\mathcal{G}^*(t)$: Instead of generating dynamic network structures directly, we proceed as follows. We generate static networks, as before, and make every edge of each network $\mathcal{G}^*$ to follow a particular edge transmission rate evolution pattern, to obtain dynamic networks $\mathcal{G}^*(t)$. Then, an edge $(i, j) \in \mathcal{E}^*(t)$ exists at time t if $\alpha_{i,j}(t) > 0$, and disappears otherwise.
(2) Cascade generation: We consider five edge evolution patterns: Slab, Square, Chainsaw, Hump and Constant (see Figure 5.12). Slab and Hump patterns model outgoing connections of sites that become popular for a short period of time. Square and Chainsaw patterns model incoming connections to sites that perform updates periodically at specific times of the day or days of the week. Constant pattern represents connections between sites that interact at any time and during a long period of time, usually large media sites. We consider Chainsaw, Hump and Constant to be examples of *Type I* pattern, without discontinuities, and Slab and Square to be examples of Type II patterm, with discontinuities. Then, we assign to each edge in the network an evolution pattern chosen uniformly at random from the set of the above five patterns. Then, we generate transmission rate values $\alpha^*_{j,i}(t)$ for each edge according to its chosen evolution pattern, using the generative model defined in Section 4.2. The evolving edge transmission rate $\alpha^*_{j,i}(t)$ models how quickly information spreads from one node to another. Finally, we generate 1,000 information cascades per time step. For each cascade we randomly pick the cascade root node. Given the node activation times from the recorded cascades, our goal then is to find the true edges of the network and for each edge discover its transmission rate evolution

pattern. In other words, inferring how each edge transmission rate $\alpha(t)$ evolves over time.
(3) Length of the sampling time window: Intuitively, the shorter the sampling time window T_s in the stochastic gradient descent implementation, the quicker our algorithm tracks changes in transmission rates in a dynamic network. However, a short sampling time window results in less reliable estimates because we sample fewer cascades. In our experiments on synthetic data, we set the time window to $T_s = 5$ time units, and we allow transmission rates to change value per time unit.

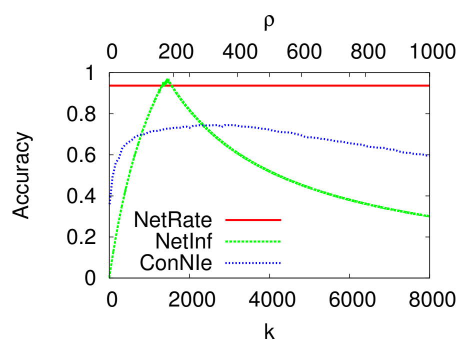
Performance in static networks

First, we compare the performance of NetRate in static weighted networks against one of our network inference algorithms for unweighted networks, NetInf, and one state of the art network inference algorithm for weighted networks, ConNIe Myers and Leskovec (2010), by comparing the inferred and true networks via three measures: precision, recall and accuracy. Precision is the fraction of edges in the inferred network $\hat{\mathcal{G}}$ present in the true network $\mathcal{G}^*$. Recall is the fraction of edges of the true network $\mathcal{G}^*$ present in the inferred network $\hat{\mathcal{G}}$. 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. Second, we evaluate how accurately NetRate infers transmission rates over edges by computing the normalized mean absolute error (MAE), $E\big[|\alpha^* - \hat{\alpha}|/\alpha^*\big]$, where α^* is the true transmission rate and $\hat{\alpha}$ is the estimated transmission rate.

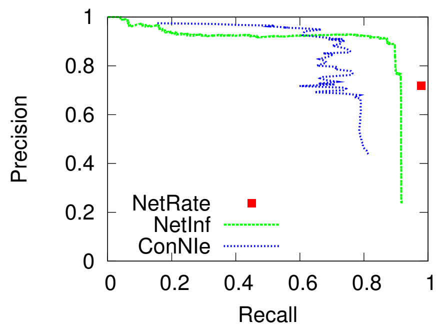
Figure 5.4 compares the precision, recall and accuracy of NetRate with NetInf and ConNIe for two types of static Kronecker networks: hierarchical community structure with exponential model for 5,000 cascades and random with Rayleigh model for 2,000 cascades, and a static Forest Fire network with power law model for 5,000 cascades over an observation window of length $T = 10$. In terms of precision-recall, NetRate outperforms ConNIe and NetInf for all the synthetic examples in the Pareto sense (Boyd and Vandenberghe, 2004). More specifically, if we set ConNIe and NetInf's tunable parameters to provide solutions with the same precision as



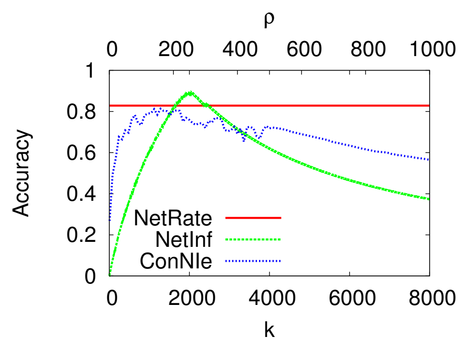
(a) Precision-recall (Hierarchical, Exp)



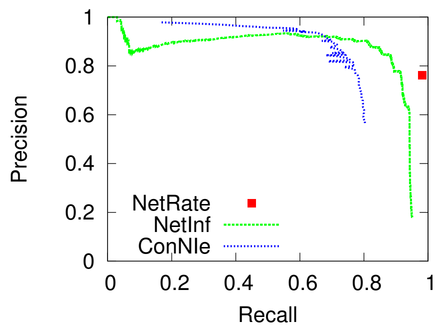
(b) Accuracy (Hierarchical, Exp)



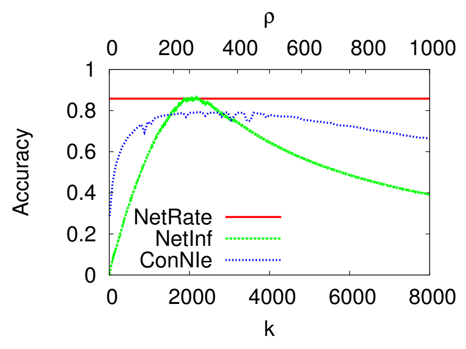
(c) Precision-recall (Random, Ray)



(d) Accuracy (Random, Ray)



(e) Precision-recall (Forest fire, Pow)



(f) Accuracy (Forest fire, Pow)

Figure 5.4: Panels (a,c,e) plot precision against recall; panels (b,d,f) plot accuracy. For ConNIe and NetInf we sweep over parameters ρ (penalty factor) and k (number of edges) respectively to control the solution sparsity in both algorithms, thereby generating a family of inferred models. NetRate has no tunable parameters and therefore yields a unique solution. (a,b): 1,024 node hierarchical Kronecker network with exponential model for 5,000 cascades. (c,d): 1,024 node random Kronecker network with Rayleigh model for 2,000 cascades. (e,f): 1,024 node Forest Fire network with power law model for 5,000 cascades.

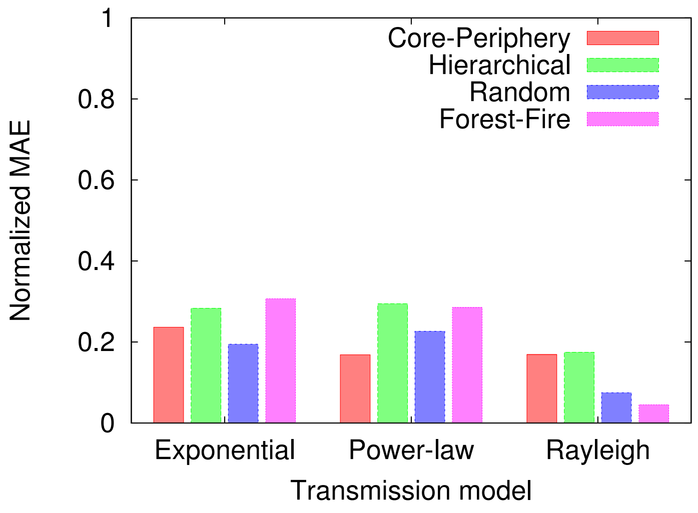
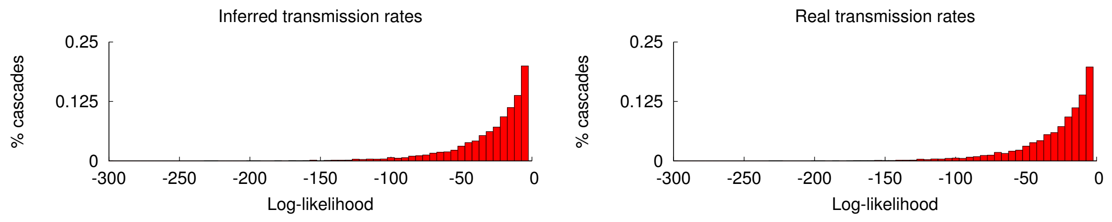


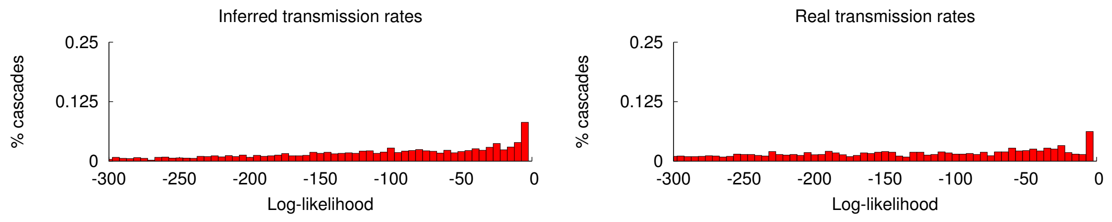
Figure 5.5: Normalized mean absolute error (MAE) of NetRate for three types of Kronecker networks (1,024 nodes and 2,048 edges) and a Forest Fire network (1,024 edges and 2,422 edges) for 5,000 cascades. We consider three models of transmission likelihoods: exponential (Exp), power-law (Pow) and Rayleigh (Ray).

NetRate, NetRate's recall is always higher than the other two methods. Strikingly, ConNIe and NetInf do not achieve NetRate's recall for any precision value. NetRate outperforms ConNIe with respect to accuracy for any penalty factor ρ in all the synthetic examples. It is also more accurate than NetInf for most values of k (number of edges). Importantly, NetInf and ConNIe yield a curve of solutions from which have to select a point blindly (or at best heuristically), whereas NetRate yields a unique solution without any tuning. However, considering that NetRate and ConNIe have more degrees of freedom than NetInf, it is surprising how well NetInf performs in comparison with them despite assuming uniform transmission rates and priors.

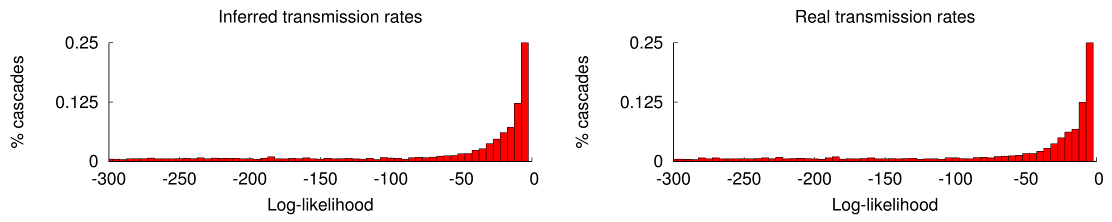
Figure 5.5 shows the normalized MAE of the estimated transmission rates for the same networks, computed on 5,000 cascades. The normalized MAE is under 25% for almost all networks and transmission models – surprisingly low given we are estimating more than 2,000 non-zero real numbers.



(a) Hierarchical, Exp

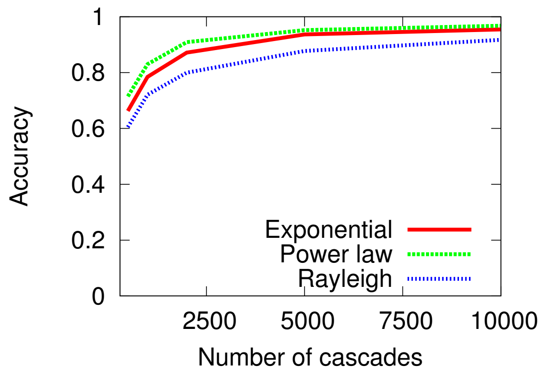


(b) Random, Ray

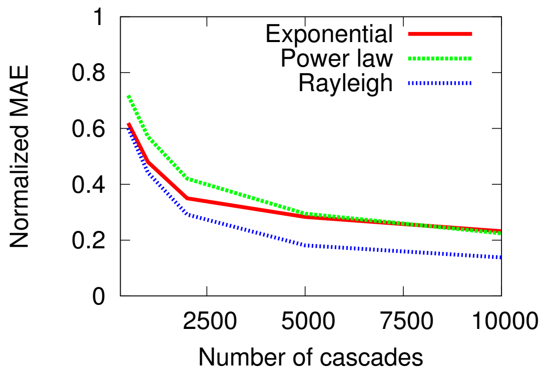


(c) Forest-Fire, Pow

Figure 5.6: Distribution of the log-likelihood of the cascades for (a) 5,000 cascades in a hierarchical Kronecker network (1,024 node, 2,048 edges) with exponential model, (b) 2,000 cascades in a random Kronecker network (1,024 node, 2,048 edges) with Rayleigh model and (c) 5,000 cascades in a Forest Fire network (1,024 edges and 2,422 edges) with power law model over an observation window of length $T = 10$. We compare the log-likelihoods of the cascades for true networks and inferred networks. All networks are static.



(a) Accuracy



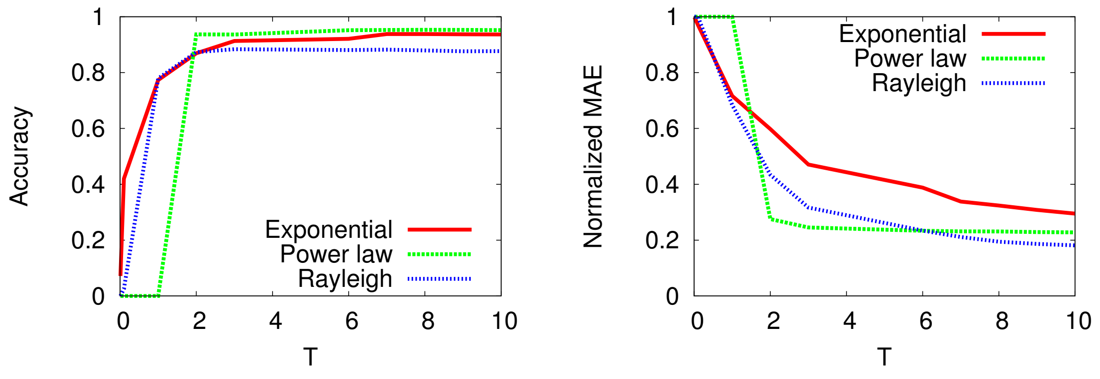
(b) Normalized MAE

Figure 5.7: Performance of NetRate vs. amount of cascade data
Performance of NetRate vs. cascade coverage for a static hierarchical Kronecker network (1,024 nodes and 2,048 edges) with exponential, power-law and Rayleigh transmission models over an observation window of length $T = 10$.

Solution quality

Given a diffusion network, we may expect that some cascades are more likely than others. Moreover, we would like that NetRate outputs inferred networks that produce the same cascades' likelihoods as the ones given by the true networks. Therefore, we now compare the log-likelihood per cascade for the true networks and the inferred networks for different networks and transmission model.

Figure 5.6 plots the distribution of log-likelihoods of the set of cascades that we used for network inference in the previous section. We compute the distribution of the log-likelihoods of the cascades for true networks and inferred networks. We observe that the distribution of log-likelihoods across cascades depends on the type of network and transmission model. Both the hierarchical Kronecker with exponential model and the Forest Fire with power law model result in many cascades having a high likelihood, specially in the case of the Forest Fire with power law model, and a rapid decay of the number of cascades with the log-likelihood value. In contrast, the random Kronecker with Rayleigh model produce a set of cascades with log-likelihood values covering uniformly a much wider range. The distribution of the log-likelihoods of the cascades are always very similar for the real and the inferred networks.



(a) Accuracy (b) Normalized MAE

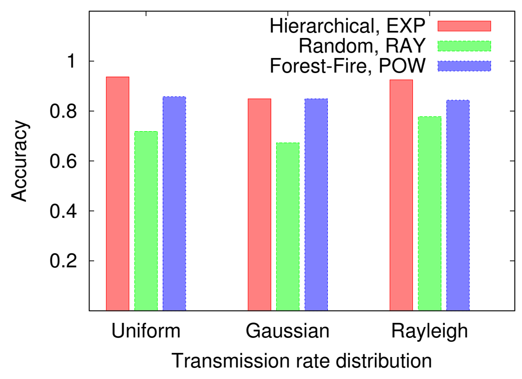
Figure 5.8: Performance of NetRate vs. time horizon for a static hierarchical Kronecker network (1,024 nodes and 2,048 edges) with exponential, power-law and Rayleigh transmission models.

Performance vs. cascade coverage

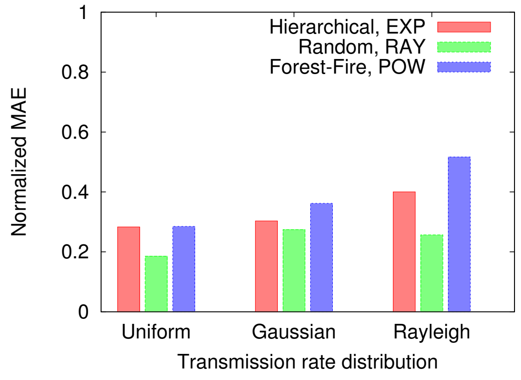
Observing more cascades leads to higher precision-recall and more accurate estimates of the transmission rates. Figure 5.7 plots the accuracy and normalized MAE of the estimated transmission rates against the number of observed cascades for a static hierarchical Kronecker network with three transmission models over an observation window of length $T = 10$. Estimating transmission rates is considerably harder than simply discovering edges and therefore more cascades are needed for accurate estimates. As many as 5,000 cascades are required to obtain normalized MAE values lower than 20%. Up to 5,000 cascades, the normalized MAE decreases quickly as a function of the number of cascades. Beyond 5,000 cascades, it becomes more difficult to decrease further the normalized MAE by adding cascades.

Performance vs. time horizon

Intuitively, the longer the observation window, the more accurately NetRate infers transmission rates. Figure 5.8 confirms this intuition by showing the accuracy and normalized MAE of the estimated transmission rates for different time horizons T for a static hierarchical Kronecker with exponential, power-law and Rayleigh transmission



(a) Accuracy



(b) Normalized MAE

Figure 5.9: Performance of NetRate vs. transmission rate distribution. Panels plot (a) accuracy and (b) normalized MAE of the estimated transmission rates against the transmission rate distribution for a hierarchical Kronecker network (1,024 nodes and 2,048 edges) with exponential model for 5,000 cascades, a random Kronecker network (1,024 nodes and 2,048 edges) with Rayleigh model for 2,000 cascades and a Forest Fire network (1,024 edges and 2,422 edges) with power law model for 5,000 cascades over an observation window of length $T = 10$. All networks are static.

models for 5,000 cascades. The longer is the time horizon T, the weaker is the *right-censoring* in the diffusion data and the more accurately NetRate infers the transmission rates. However, once we reach a sufficiently long time horizon T, further increasing the recording time does not increase the performance significantly since there are not unrecorded activations anymore.

Performance vs. transmission rate distribution

We have carried out experiments using synthetic networks in which the transmission rates of the edges are always drawn from a uniform distribution. Since this assumption may be often violated in real networks we now consider networks in which we set the transmission rates of the edges by drawing samples from (i) a uniform distribution, (ii) a Gaussian distribution ($\mu = 0.5$, $\sigma = 0.5$; we reject any negative samples) and (iii) a Rayleigh distribution ($\sigma = 0.25$).

Figure 5.9 plots accuracy and normalized MAE of the estimated transmission rates against the transmission rate distribution for a static hierarchical Kronecker network

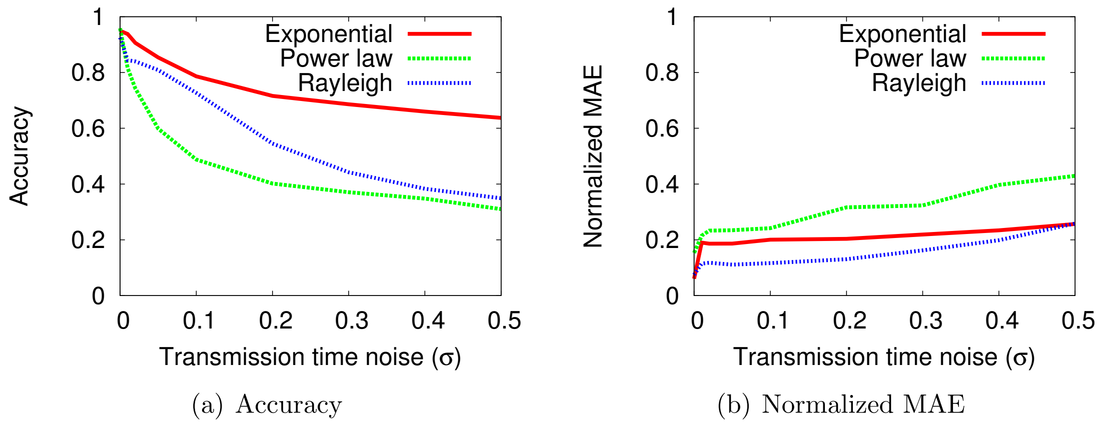


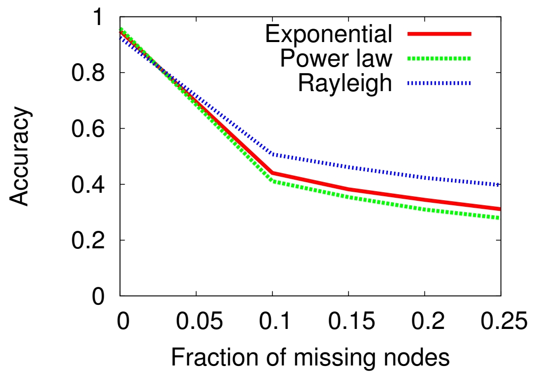
Figure 5.10: Performance of NetRate vs. amount of additive Gaussian noise (standard deviation σ) in the transmission times for a static random Kronecker network (1,024 nodes and 2,048 edges) with exponential, power-law and Rayleigh transmission models over an observation window of length $T = 10$.

with exponential model for 5,000 cascades, a static random Kronecker network with Rayleigh model for 2,000 cascades and a static Forest Fire network with power law model for 5,000 cascades over an observation window of length $T = 10$. In all networks, the accuracy remains relatively stable across transmission rate distributions. However, the more skewed the transmission rate distribution is, the greater is the normalized MAE (*i.e.*, it is easier to estimate transmission rates drawn from a uniform distribution than from a Gaussian or Rayleigh).

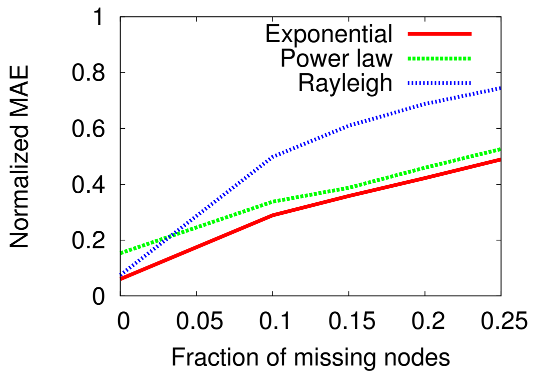
Performance vs. transmission time noise

When we work with real data, it may happen that the true pairwise transmission likelihoods differ from the parametric models we assume or that the observed activation times may have been corrupted by noise. We then study the accuracy and normalized MAE of NetRate as a function of the noise of the transmission times between activations. To this end, we add Gaussian noise to the transmission times between activations in the cascade generation process.

Figure 5.10 shows the accuracy and normalized MAE against the amount of



(a) Accuracy



(b) Normalized MAE

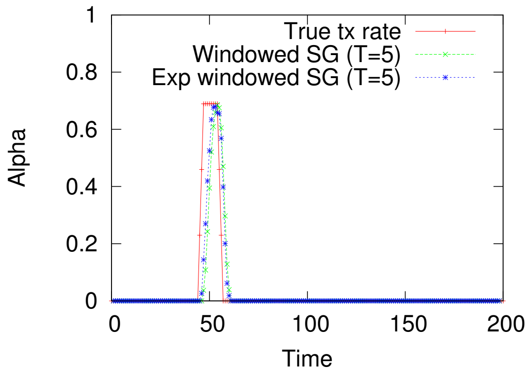
Figure 5.11: Performance of NetRate vs. fraction of missing nodes per cascade for a static random Kronecker network (1,024 nodes and 2,048 edges) with exponential, power-law and Rayleigh transmission models over an observation window of length $T = 10$.

Gaussian noise added to the transmission times between activations for a static random Kronecker network with exponential, power-law and Rayleigh transmission models for 5,000 cascades. In all three transmission models, the normalized MAE (*i.e.*, transmission rate inference) is more robust against noise than the accuracy (*i.e.*, network structure inference).

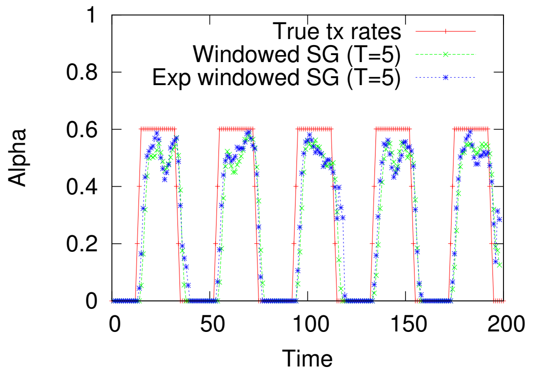
Performance vs. missing activations

In many real world scenarios, we do not observe all nodes that become activated during the observation window. For example, media sites and blogs may publish contents that only subscribers can read and members of a social network can restrict the visibility of certain posts. Therefore, we consider collections of cascades where a random fraction of each cascade is missing. This means that we first generate a set of cascades, but then only record node activation times of a fraction of nodes.

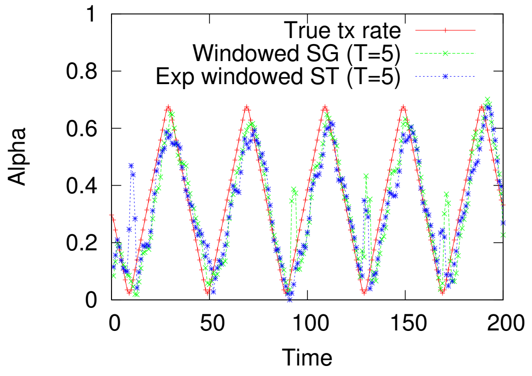
Figure 5.11 shows the accuracy and normalized MAE against the fraction of missing nodes per cascade for a static random Kronecker network with exponential, power-law and Rayleigh transmission models for 5,000 cascades. Missing data degrades the performance of NetRate significantly more than noise. Although there



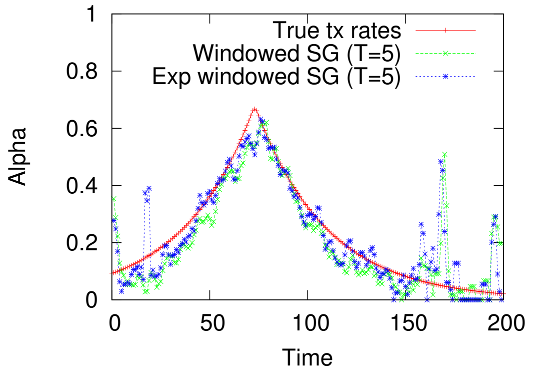
(a) Slab



(b) Square



(c) Chainsaw



(d) Exponential

Figure 5.12: True and inferred transmission rate over time for edges with different transmission rate trends for a 512 node, 1,024 edge core-periphery Kronecker network with exponential model for 200 time units with 1,000 cascades per time unit. Our method is able to track the changing transmission rate values over time. It works better when the transmission rate trend is continuous (c,d) than when there is a discontinuity (a,b).

has been increasing effort devoted to correcting for missing data in information cascades, previous algorithms attempt to output cascades with the same structural properties of the original (complete) cascades from the incomplete cascades (Sadikov et al., 2011) or to simply estimate the cascade width and length (Chierichetti et al., 2011) but the inferred cascades may be actually very different to the original cascades. It remains an open problem how to correct for missing data in the context of network inference.

An interesting open question is whether *localized* missing observations can be detected. For example, is it possible to detect when certain memes are suppressed on specific websites or in specific regions?

Performance in dynamic networks

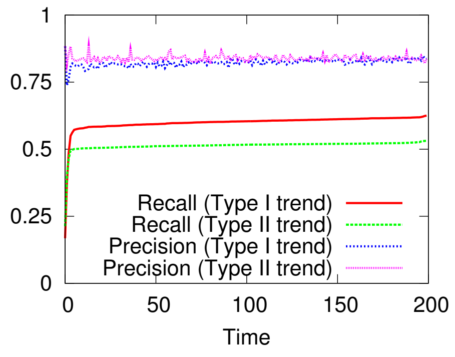
In this section, we evaluate the performance on NetRate in dynamic networks. We first show qualitatively how our algorithm performs for different transmission rate trends, and then evaluate quantitatively its performance.

Figure 5.12 shows the true and inferred transmission rates for four different edges, each with a different evolution pattern: Slab, Square, Chainsaw and Humb, in a 512 node, 1,024 edge core-periphery Kronecker network with 20% of the edges following each of the five rate trends. We generated and recorded an average of 1,000 cascades per time unit using an exponential pairwise transmission model. Our method is able to track the evolving edge transmission rate over time for all evolution patterns. It gives near perfect performance when edge transmission rate evolves continuously (Chainsaw, Hump). Interestingly, even when the edge transmission rate evolves discontinuously (Slab, Square), InfoPath manages to track it.

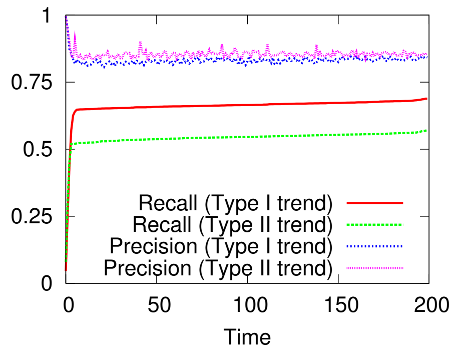
Now, we compute four different measures: Precision, Recall and Accuracy of inferred edges as well as Mean Squared Error (MSE) in the edge transmission rate, in order to evaluate the performance of our algorithm quantitatively. Precision at time t is the fraction of edges in the inferred network $\hat{\mathcal{G}}(t)$ present in the true network $\mathcal{G}^*(t)$. Recall at time t is the fraction of edges of the true network $\mathcal{G}^*(t)$ present in the inferred network $\hat{\mathcal{G}}(t)$. Accuracy at time t is defined as

$$1 - \frac{\sum_{i,j} |I(\alpha^*_{i,j}(t)) - I(\hat{\alpha}_{i,j}(t))|}{\sum_{i,j} I(\alpha^*_{i,j}(t)) + I(\hat{\alpha}_{i,j}(t))},$$

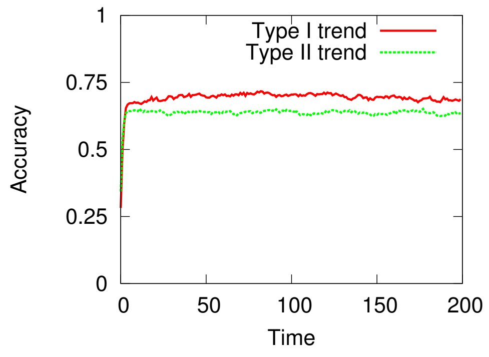
where $\alpha^*(t)$ is the true transmission rate at time t, $\hat{\alpha}(t)$ is the estimated transmission rate at time t, and $I(\alpha(t)) = 1$ if $\alpha(t) > 0$ and $I(\alpha(t)) = 0$ otherwise. Inferred networks with no edges or only false edges have zero accuracy. Last, Mean Squared Error (MSE) at time t is defined as $E\big[||\alpha^*(t) - \hat{\alpha}(t)||^2\big]$, where $\alpha^*(t)$ is the true



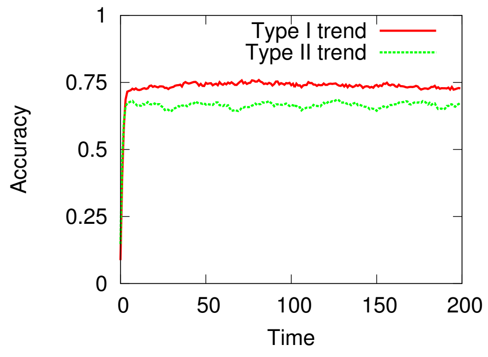
(a) P-R (C-P, Exp)



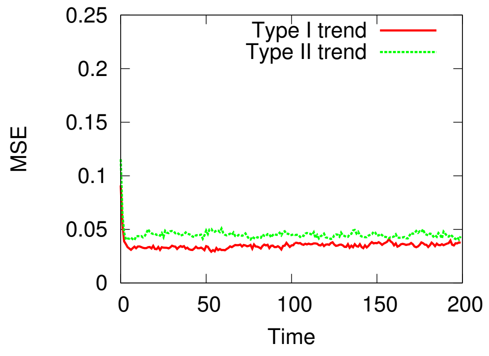
(b) P-R (HI, Ray)



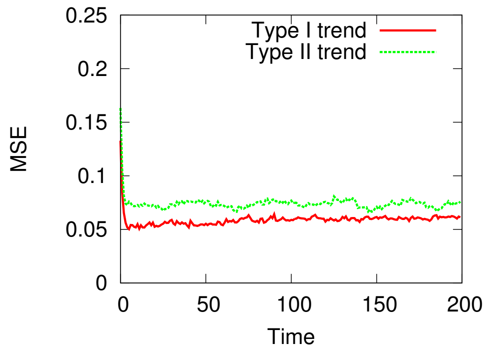
(c) Accuracy (C-P, Exp)



(d) Accuracy (HI, Ray)



(e) MSE (C-P, Exp)



(f) MSE (HI., Ray)

Figure 5.13: Precision and recall (P-R), accuracy and mean square error (MSE) of our stochastic method against time. (a,c,e): 1,024 node, 2,048 edge dynamic core-periphery (C-P) Kronecker network with exponential model, (b,d,f): 1,024 node, 2,048 edge dynamic hierarchical (HI) Kronecker network with Rayleigh model. In both networks, type I (Chainsaw, Hump) and type II (Slab, Square) trends for transmission rates were generated, and 1,000 cascades per unit time were recorded.

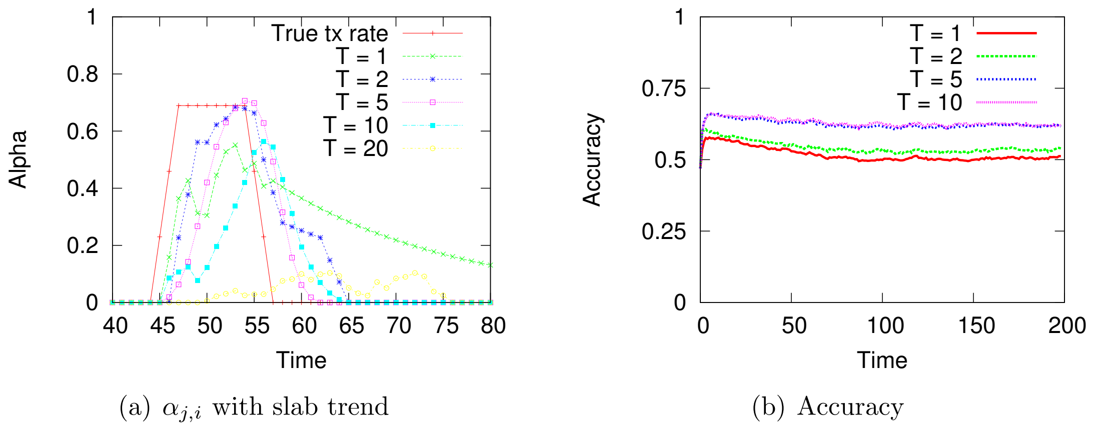


Figure 5.14: Performance vs sampling time window. Panel (a) shows the true and inferred transmission rates for a transmission rate with a Slab evolution pattern for different sampling time window lengths. Panel (b) shows accuracy across time for different sampling time window length for a 512 node, 1,024 edge dynamic core-periphery Kronecker network. Half the edges have transmission rates that follow a Slab evolution pattern, and half the edges have a constant transmission rate. We generated and recorded an average of 1,000 cascades per time unit using an exponential pairwise transmission model.

transmission rate at time t and $\hat{\alpha}(t)$ is the estimated transmission rate.

Figure 5.13 shows Precision, Recall, Accuracy, and MSE over time for two 1,024 node, 2,048 edge dynamic kronecker networks, core-periphery (parameter matrix $[0.9, 0.5; 0.5, 0.3]$) and hierarchical (Clauset et al., 2008) ($[0.9, 0.1; 0.1, 0.9]$), with exponential and Rayleigh pairwise transmission models respectively. We generated continuous (Chainsaw, Hump) and discontinuous (Slab, Square) evolution patterns for transmission rates, $\alpha^*_{j,i}(t) \in [0, 1]$ for all t, and we recorded 1,000 cascades per unit time. The performance of our method is stable across time, and as noticed qualitatively, continuous trends are easier to track and estimate than discontinuous ones.

Performance vs. sampling time window

Intuitively, the shorter the sampling time window T_s in the stochastic gradient descent implementation, the quicker our algorithm tracks changes in transmission rates in a dynamic network. However, a short sampling time window results in less reliable estimates because we sample fewer cascades.

Figure 5.14(a) shows the true and inferred transmission rates for a transmission rate which evolves as a Slab for different sampling time window lengths. The experimental results support the intuition. We observe that the shorter the sampling time window, the quicker we are able to track the step up. However, when the sampling time window is too short, stochastic gradient descend do not sample cascades with activations of the source of the edge and the rate decays only by aging.

Figure 5.14(b) shows accuracy across time for different sampling time window length for a 512 node, 1,024 edge dynamic core-periphery Kronecker network. Half the edges have transmission rates that evolve as a Slab, and half the edges have a constant transmission rate. We generated and recorded an average of 1,000 cascades per time unit using an exponential pairwise transmission model. Too short or too long sampling time windows result in lower accuracy.

5.3.5 Experiments on real data

In this section we analyze dynamic networks based on real diffusion data, since information pathways change over time, depending upon the information content that propagate through them (Romero et al., 2011a; Myers et al., 2012). For example, a real world event may occur for a limited period of time and thus news related to the event spread quicker and to larger parts of the network around such time period. At any given time, there are many different real world events, topics, and content that propagates through the Web, leading to different emerging and vanishing information pathways, and thus an underlying dynamic network. In order to better understand these temporal changes, we aim to reconstruct dynamic networks and the information pathways for particular real world events and topics. All the data, code and additional results are available at the supporting websites (NetRate, 2011; InfoPath, 2013).

Dataset description

We use the topic-based MemeTracker dataset, which contains more than 300 million news articles and blog posts from 3.3 million online sources over a period of one year, from March 2011 till February 2012[3]. Based on this raw data, we use two different methodologies to trace information on the Web and then create two different datasets: hyperlink cascade dataset and MemeTracker cascade dataset. We refer the reader to Section 2.5.3 for more details on the topic-based MemeTracker dataset and an in-depth description of the two methodologies we used to trace information on the Web.

Our aim is to consider sites that actively spread memes over the Web. We achieve this by selecting top 5,000 sites in terms of the number of memes they mentioned. Moreover, we are interested in inferring dynamic networks related to particular topics or events. So, we assume we are also given a keyword query Q related to the event/topic of interest. When inferring a network for a given query Q, we only consider documents (and the memes they mention) that include keywords Q. Then, we build information cascades using only those memes and apply our algorithm to infer the edges and evolving edge transmission rates. The edge transmission rates explain the propagation of information related to a given topic or real world event Q. For each query Q we infer one network per day. Table 2.2 summarizes the number of sites and meme cascades for several topics and real world events.[4].

Implementation and scalability

We developed an efficient distributed implementation of our algorithm using stochastic gradient descend in C++, which uses the graph library SNAP (SNAP, 2012). We deployed the implementation in a cluster with 1000 CPU cores and 6 TB of RAM. With this setup, we inferred 38 dynamic networks, one per topic or news world event, with a daily resolution for a period of one year from March 2011 to February 2012, with thousands of nodes using hundreds of thousands of cascades in less than 4 hours.

[3]Data available at the `http://snap.stanford.edu/infopath/`

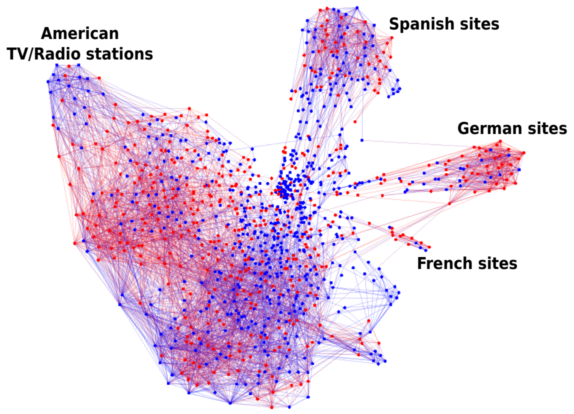
[4]Additional dynamic diffusion networks for other topics and news events are available at the supporting website (InfoPath, 2013)

Note that inferring 38 dynamic with a daily resolution for a one year period is equivalent to solving Eq. 5.12 more than 13,000 times (38 x 365) for millions of pairwise transmissions transmission rates. We also tested our algorithm on larger datasets. For example, for "Occupy Wall Street movement", we were able to infer a 43,415-node dynamic network over a period of 18 months, from January 2011 to June 2012, using 1,381,793 information cascades.

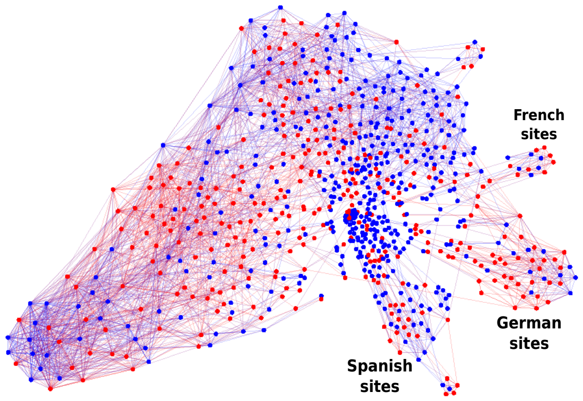
Visualizing the information pathways

Figures 5.15, 5.16 and 5.17 plot diffusion networks for three different 2011 world events: Fukushima nuclear disaster, UK royal wedding, and civil uprise in Syria. Each network is shown at three different time points. Red nodes represent mainstream media sites, and blue nodes represent blogs (Leskovec et al., 2009).

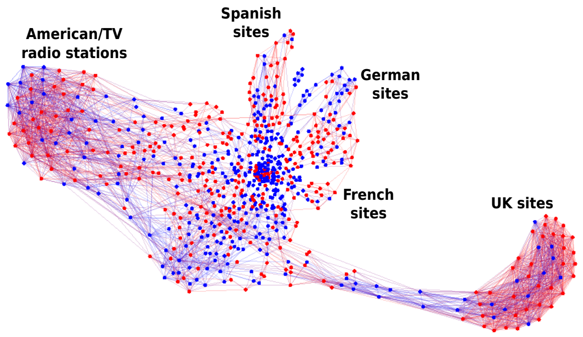
Based on the figures, we draw several interesting observations. Most often, information propagates through a core-periphery network structure. Such structure emerges by few central media sites and blogs driving the adoption of memes across the Web (Gomez-Rodriguez et al., 2010). However, the network structure often changes dramatically over time, and we find clusters that emerge and vanish in short periods of time. For example, the information networks for Syria's uprise illustrated in Figure 5.17, do not have any clear clustering structure. However, on December 2, 2011 (Figure 5.17(c)) a cluster suddenly emerges in the network. Further investigation reveals that the cluster is composed of UK news sites and blogs that discuss recently implemented EU sanctions against Syria. Generally, it is common to observe sudden formation of clusters of sites from specific geographical areas. This is specially noticeable in the information network for Fukushima's disaster, in Figure 5.15. Such clusters often form due to language boundaries, since such boundaries prevent memes to flow across countries or continents. Moreover, we often observe that such clusters are caused by a common external event (Myers et al., 2012), like in the case of UK discussion on EU sanctions against Syria. Inferred dynamic networks can thus be used to investigate the flow of information as well as to detect external events that cause sudden perturbations to the diffusion network structure.



(a) Fukushima (2011-03-18)

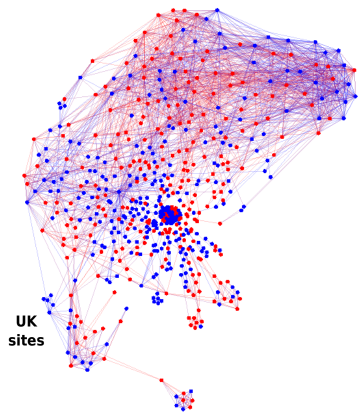


(b) Fukushima (2011-06-25)

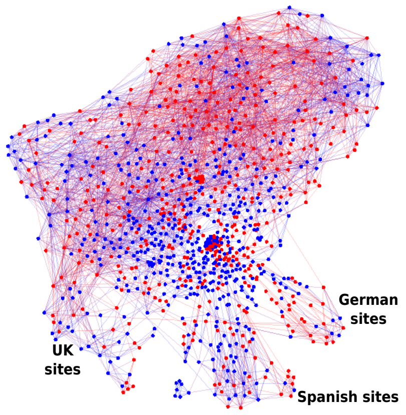


(c) Fukushima (2011-10-13)

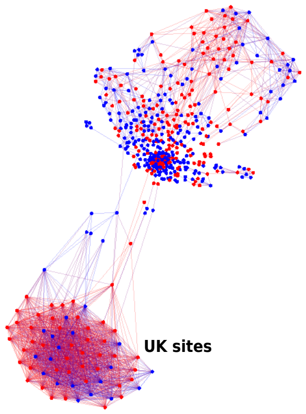
Figure 5.15: Snapshots of the dynamic diffusion networks for Fukushima at three different times. Red nodes are mainstream media, and blue nodes are blogs.



(a) UK royal wedding (2011-04-02)

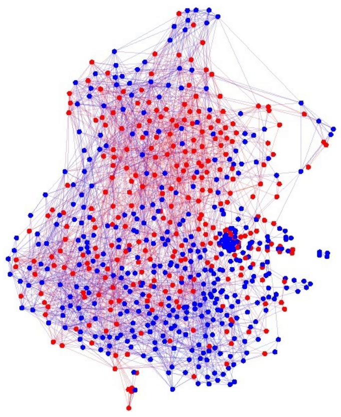


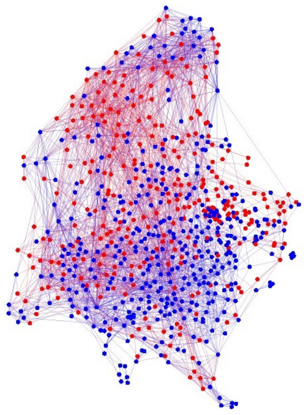
(b) UK royal wedding (2011-05-02)

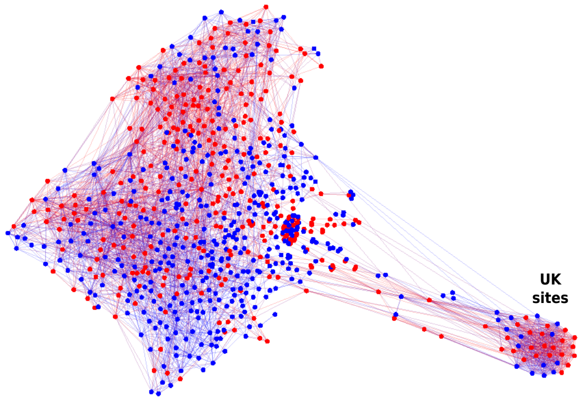


(c) UK royal wedding (2011-11-15)

Figure 5.16: Snapshots of the dynamic diffusion networks for UK royal wedding at three different times. Red nodes are mainstream media, and blue nodes are blogs.


(a) Syria's uprise (2011-04-05)


(b) Syria's uprise (2011-06-02)



(c) Syria's uprise (2011-12-02)

Figure 5.17: Snapshots of the dynamic diffusion networks for Syria's uprise at three different times. Red nodes are mainstream media, and blue nodes are blogs.

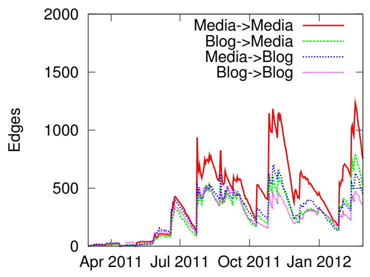
Evolution of edge transmission rates

Next, we aim to study the evolution of links among different types of sites. We label the nodes in our network as mainstream media and blog, and compute the number of links between different types of sites over time. Figure 5.18 gives the results for several inferred diffusion networks for different topics and world events. We note several interesting patterns.

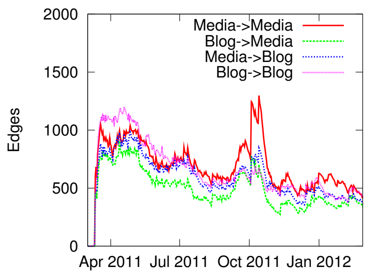
The connectivity changes tend to reflect the amount of attention that a news event or a topic triggers over time. Unexpected news events, like the sex scandal of the director of the International Monetary Fund Strauss-Kahn on May 14, 2011 in Fig. 5.18(g) or the death of British singer Amy Winehouse on July 23, 2011 in Fig. 5.18(a), result in a dramatic increase in the number of edges over a short period of time. More general topics, like the NBA in Fig. 5.18(e), result in a network with more stable connectivity over time. Certain types of news are sometimes spreading earlier among blogs than mainstream media. This is especially the case for population wide events like the Fukushima nuclear disaster, civil war in Libya and civil uprise in Syria (Fig. 5.18(b, c, h). However, it happens more frequently that the largest amount of links are mainstream media-to-mainstream media and the fewest links point from blogs to mainstream media. These results are intuitive and consistent with previous work (Gomez-Rodriguez et al., 2010; Leskovec et al., 2009) that observed most often information flows from mainstream media to blogs (and rarely the other way around). However, as we see here for population level events and social movements (like, in case of the civil unrest in the Middle East) social media plays crucial role in information dissemination and organization of civil movements.

Evolution of node centrality

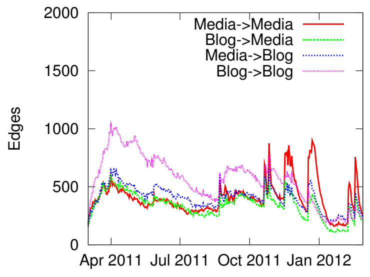
Having studied the dynamics of edges in the network we now move towards investigating the network centrality of blogs and mainstream media sites over time for different topics and world events. To measure network centrality of node S in the network at time t, we first compute shortest path length from S to any other node R in the network. Then centrality of node S is defined as $\sum_R 1/d(S,R)$, where $d(S,R)$ is the



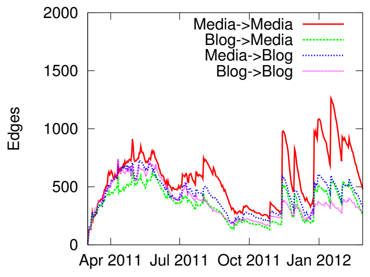
(a) Amy Winehouse



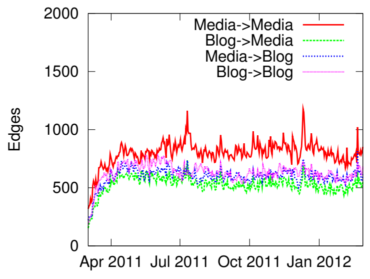
(b) Fukushima



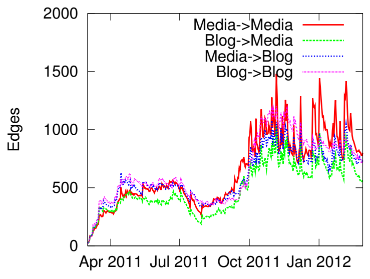
(c) Gaddafi



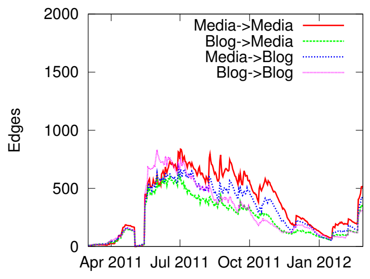
(d) UK royal wedding



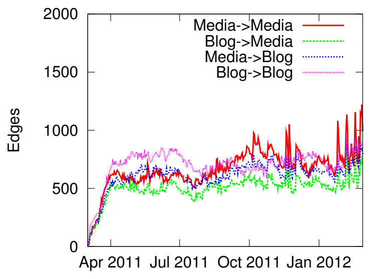
(e) NBA



(f) Occupy



(g) Strauss-Kahn



(h) Syria

Figure 5.18: Number of links that point between different types of sites across time for several inferred diffusion networks for eight different topics or 2011 world events.

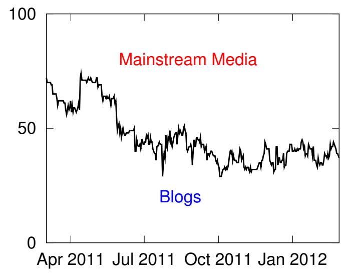
shortest path length from S to R (if R is not reachable from S then $d(S, R) = \infty$). For networks with core-periphery structure, nodes with high centrality are typically located in the "central" core of the network.

Figure 5.19 plots the percentage of blogs among the top 100 most central sites over time for eight different topics/events of 2011. Perhaps surprisingly, we observe there is a about the same number of mainstream media and blogs in the top-100 most central nodes for most networks – the number of blogs in the top-100 does not typically decreases below 30% or increases over 70%. For some topics, mainstream media are always more *central* (*e.g.*, baseball and NBA in Figures 5.19(a, b)). In contrast, for other topics, blogs dominate mainstream media over a significant amounts of time (*e.g.*, Gaddafi in Fig. 5.19(c)). Centrality of mainstream media and blogs can be relatively constant (Fig. 5.19(a,b)) or more time-varying (Fig. 5.19(c,h)). We find that a significant rise in the number of central blogs is often temporally correlated with an increasing social unrest (*e.g.*, the Occupy Wall Street movement in Sept-Nov 2011 in Fig. 5.19(f)).

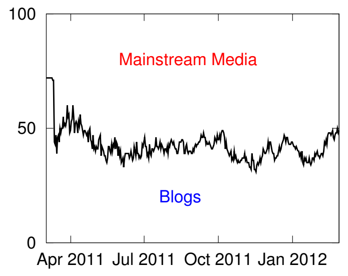
Accuracy on real data

So far, we have used *memes* to trace the flow of information over the Web and have made several qualitative observations about the structure and dynamics of information pathways in online media. We now proceed and attempt to also quantitatively evaluate our algorithm on real data. In case of real data the ground-truth information diffusion network is impossible to obtain. However, we can use the temporal dynamics of hyperlinks created between news sites as a proxy for real information flow. Thus, by observing the times when sites create hyperlinks, our goal is to infer the 'targets' of the links (i.e., infer the hyperlink network from the hyperlinks times).

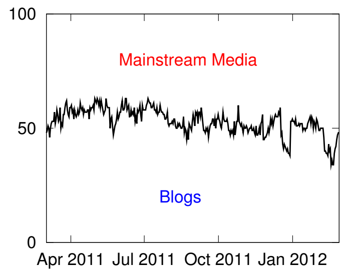
We proceed as follows. First, we discretize the time in days, we generate one network $\mathcal{G}^*(t)$ per day t, in which we add an edge (u, v) if a document on a site u linked to a document on a site v within the last day. Then, we build a set of hyperlink cascades. A hyperlink cascade c_h starts when a site publishes a piece of information and then other sites use hyper-links to refer to it. Since all our documents/posts are time stamped, we can trace the hyperlinks in the reverse direction and obtain



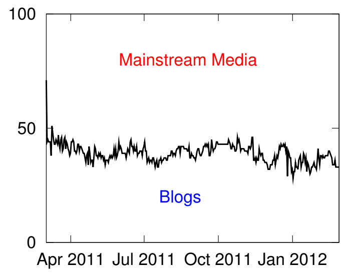
(a) Amy Winehouse



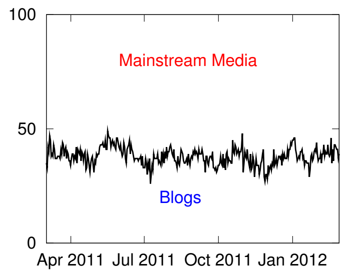
(b) Fukushima



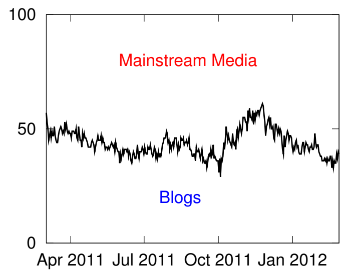
(c) Gaddafi



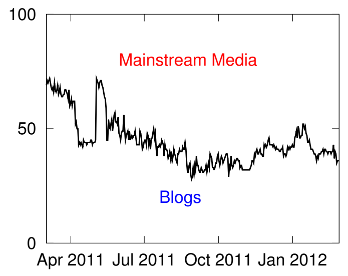
(d) UK royal wedding



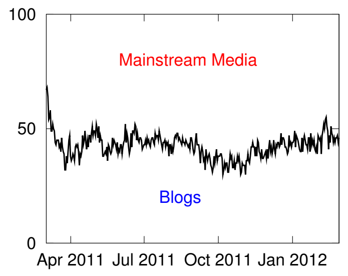
(e) NBA



(f) Occupy

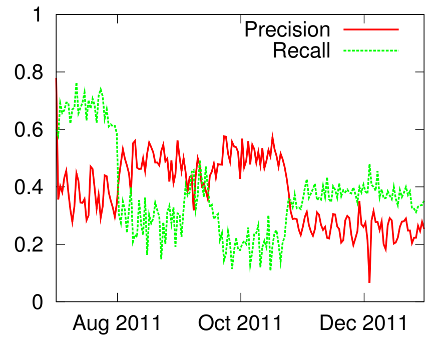


(g) Strauss-Kahn

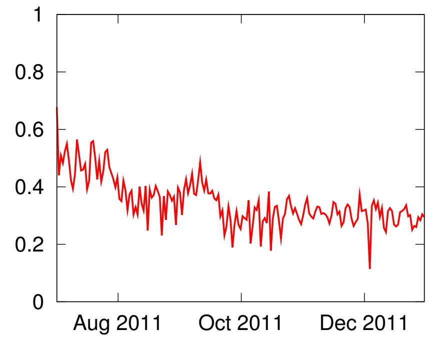


(h) Syria

Figure 5.19: Percentage of blogs and mainstream media in top-100 most *influential* sites for eight different topic or 2011 world event inferred diffusion networks. Mainstream media are represented in red, and blogs in blue.



(a) Precision-Recall



(b) Accuracy

Figure 5.20: Precision, recall, and accuracy of our stochastic method against time for a dynamic hyperlink network with 11,461 nodes and 19,915 total number of edges across time, using 495,655 hyperlink cascades from July 2011 to December 2011.

information cascades. We extracted almost 0.5 million hyperlink cascades from 3.3 million websites from July 2011 till December 2012. Our aim is to use the hyperlink cascades to infer the dynamic network $\mathcal{G}^*(t)$. We then evaluate how many edges our algorithm estimates correctly by computing Accuracy, Precision and Recall for each day.

Figure 5.20 shows Precision, Recall, and Accuracy over time for a dynamic hyperlink network with 11,461 nodes and 19,915 edges created over time, using 495,655 hyperlink cascades from July 2011 to December 2011. We assume an exponential edge transmission model. We observe weekly periodicity and the overall encouraging performance of around 0.4 to 0.5 for all three performance metrics.

5.4 Summary

We have developed a flexible model of the temporal structure underlying diffusion processes over weighted static and dynamic networks. The model makes minimal assumptions about the physical, biological or cognitive mechanisms responsible for diffusion. Instead, fitting the model reduces to inferring transmission rates between nodes of a network by finding the rates that maximizes the likelihood of the observed data – temporal traces left by cascades of activations. Qualitative assumptions about activations (*e.g.*, are they long-tailed or faddish?) determine the choice of parametric model on the edges. The model allows to mix exponential, power law, Rayleigh or other models, including multimodal likelihoods (Du et al., 2012), within a single inference algorithm. This provides tremendous flexibility in fitting real data which may combine long-tailed, faddish and other qualitative behaviors.

Remarkably, introducing continuous temporal dynamics, allowing variable transmission rates across edges, and avoiding further assumptions dramatically simplifies the problem compared our approach to the network inference problem for unweighted networks presented in Chapter 4. The model's parameters have natural interpretations, and it leads to a well-defined, convex maximum likelihood problem that can be solved efficiently. Importantly, we do not need to hand tune parameters to control the sparsity of the inferred network (*i.e.*, number of edges to infer or penalty terms). Indeed, heuristic l_1-like penalty terms, such as the ones used in Myers and Leskovec (2010), are unnecessary since the probabilistic model naturally imposes sparse solutions. Importantly, other research problems, as the influence maximization problem, described in Chapter 6, also get simplified under our continuous time model of diffusion.

We evaluated NetRate on a wide range of synthetic diffusion networks – both static and dynamic – with heterogeneous temporal dynamics which aim to mimic the structure of real-world social and information networks. NetRate provides a unique solution to the network inference problem with high recall, precision and accuracy. A direct comparison with the current state of the art in synthetic networks is difficult, since these methods include a parameter controlling the sparsity of the inferred network that requires blind tuning. Nevertheless, NetRate is typically better in terms of accuracy than previous methods across the full range of their tunable parameters. In addition, NetRate accurately estimates transmission rates, which other methods cannot estimate at all. The performance of ConNIe appears significantly worse than reported in Myers and Leskovec (2010); a possible explanation for the degradation is that in our work, we consider networks with heterogeneous temporal dynamics. It is surprising how well NetInf, described in Chapter 4, performs in comparison with NetRate despite assuming uniform temporal dynamics and priors. Additionally, we showed that NetRate is able to track changes in the topology of dynamic networks and provide on-line accurate estimates of the time-varying transmission rates.

Importantly, we run our algorithm on real data from online media and study how real networks and information pathways evolve over time. We found that information pathways over which general recurrent topics propagate remain relatively stable across time. In contrast, unexpected events lead to dramatic changes on the information pathways. We observed that clusters of mainstream news and blogs often emergence and vanish in matter of days. We discovered that there is an early greater increase in information transfer among blogs than among mainstream for news involving an increasing dramatic civil unrest, as the Libyan civil war, Egypt's revolution or the Syrian uprising. Finally, although we found that the amount of mainstream media and blogs among the most influential nodes for most topics or news events are comparable, the number of influential blogs on some topics or news events grows when there exists an increasing social unrest (*e.g.*, the Occupy Wall Street movement in Sept-Nov 2011).

Chapter 6

Influence maximization in diffusion networks

6.1 Introduction

This chapter presents InfluMax, a method for maximizing influence on information spread on static weighted diffusion networks (Gomez-Rodriguez and Schölkopf, 2012a), which build on the fully continuous time model of diffusion for weighted networks introduced in Chapter 4.

In more detail, we first describe how, given a set of source nodes, we can compute the average total number of activated nodes analytically using continuous time Markov chains (CTMCs). Later, we show that finding the optimal influential set of source nodes in the continuous time influence maximization problem is a NP-hard problem. We then provide an *approximation algorithm* that finds a suboptimal set of source nodes with *provable guarantees* in terms of the average total number of activated nodes.

Our results on synthetic weighted diffusion networks show that InfluMax is remarkably stable across different network topologies and transmission rates distributions. It outperforms state of the art methods in terms of influence (*i.e.*, average number of infected nodes) for different network topologies, transmission rates distributions, time horizons and source set sizes. InfluMax typically gives an influence

Symbol	**Description**
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Directed network with node set V and edge set E
$(\mathcal{G}, \mathbf{A})$	Diffusion network: directed network $\mathcal{G}$ and transmission rates $\mathbf{A}$
$\mathbf{A} = [\alpha_{i,j}]$	Pairwise transmission rates for all pair of nodes (i, j)
$\alpha_{i,j}$	Pairwise transmission rate of edge (i, j)
$f(t_j \| t_i, \alpha_{i,j})$	Pairwise transmission likelihood of edge (i, j)
$F(t_j \| t_i, \alpha_{i,j})$	Cumulative density function of edge (i, j)
T	Time horizon
$\mathcal{S}$	Node source set
s_i	Source node i
$N(\mathcal{S}; T)$	Number of activated nodes at time t given a node source set $\mathcal{S}$
$\sigma(\mathcal{S}; T) = \mathbb{E}N(\mathcal{S}; T)$	Influence function
n	Sink node
$S_n(\mathcal{B})$	Set of nodes dominated by B with respect to a sink node n
Ω_n^*	Set of self dominant node sets with respect to a sink node n
$\mathcal{I}(t \| \mathcal{S})$	Activated node set at time t given $\mathcal{S}$
$\mathcal{U}_n(t \| \mathcal{S})$	Useless node set at time t given $\mathcal{S}$ and n
$\mathcal{X}_n(t \| \mathcal{S})$	Set of disable nodes at time t given $\mathcal{S}$ and n

Table 6.1: Table of symbols for Chapter 6.

gain of $\sim 25\%$ and it achieves the greatest improvement for small time horizons; in such scenarios, considering heterogeneous transmission rates play a dramatic role. We also evaluate InfluMax on two real diffusion networks that we inferred from the MemeTracker dataset (refer to Section 2.5.2), using NetRate, described in Chapter 4. Again, InfluMax drastically outperformed the state of the art by $\sim 30\%$.

The remainder of the chapter is organized as follows: in Section 6.2, we revisit the continuous time model of diffusion over weighted networks introduced in Chapter 4 and state the continuous time influence maximization problem. In Section 6.3, we first introduce our influence maximization method, InfluMax, describing how the method evaluates and maximizes influence, and we then evaluate our method on synthetic and real diffusion networks. We conclude with a summary of our results in Section 6.4.

6.2 Problem formulation

In this section, we build on the fully continuous time model of diffusion for weighted networks introduced in Chapter 4. We start by describing how the diffusion model accounts for pairwise interactions and then continue discussing some basic implicit assumptions about diffusion processes. We conclude with a statement of the continuous time influence maximization problem.

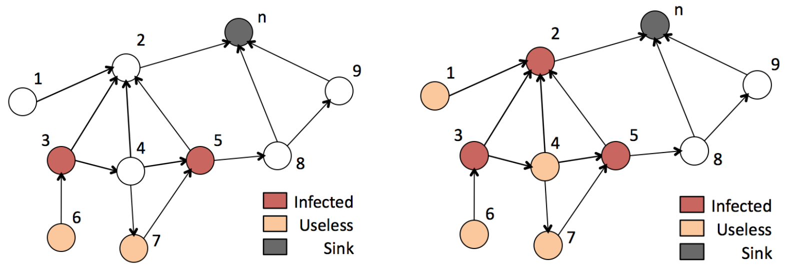
6.2.1 Pairwise interactions

We describe the pairwise interactions between nodes using three concepts, as previously in Section 4.2: pairwise transmission rates $\alpha_{i,j}$, prior probabilities of transmission $\beta_{i,j}$, and pairwise transmission likelihoods $f(t_i|t_j, \alpha_{i,j})$. The transmission rate $\alpha_{i,j}$ of an edge $(i, j) \in \mathcal{E}$ quantifies how frequently any contagion spreads from node i to node j or, in other words, the *latency* of the edge (i, j). The prior transmission probability $\beta_{i,j}$ of an edge (i, j) quantifies the probability that a contagion would eventually spread from node i to node j for arbitrarily large t_j. Finally, the pairwise transmission likelihood $f(t_j|t_i; \alpha_{i,j})$ of an edge $(i, j) \in \mathcal{E}$ is the conditional likelihood of transmission from node i, activated at time t_i, to node j. We refer the reader to Section 2.3 for an in-depth discussion of all three concepts. Now, we highlight the key assumptions on the pairwise interactions to tackle the influence maximization problem:

First, we assume networks to be weighted, we account for the general case of heterogeneous pairwise transmission rates, *i.e.*, activations can occur at different transmission rates over different edges of a network.

Second, for any contagion c, we observe activations up to a finite time horizon $T^c \to \infty$. Then, given a node i, activated at t_i, and a transmission rate $\alpha_{i,j} > 0$, the probability of survival of node j up to the time horizon T^c will be always greater than 0, $S(t_j|t_i; \alpha_{i,j}) = \int_{T^c}^{\infty} f(t_j|t_i; \alpha_{i,j}) > 0$. Then, we can assume, for simplicity, the prior probability of transmission $\beta_{i,j}$ to be 1 and yet not always observe node j to get activated before T^c.

Third, we consider exponential transmission likelihoods $f(t_j|t_i; \alpha_{i,j})$. Importantly,



(a) t_1: $|\mathcal{I}| = 2$, $|\mathcal{U}_n| = 2$, $|\mathcal{X}_n| = 4$

(b) t_2: $|\mathcal{I}| = 3$, $|\mathcal{U}_n| = 4$, $|\mathcal{X}_n| = 7$

{3 5 6 7}
{3 5 6 7 8}
{3 5 6 7 8 9}
{3 5 6 7 4}
{3 5 6 7 4 8}
{3 5 6 7 4 8 9}
{3 5 6 7 4 2 1}
{3 5 6 7 4 2 1 8}
{3 5 6 7 4 2 1 8 9}
{3 5 6 7 4 2 1 8 9 n}

(c) $X \in \Omega_n^* : A \subseteq X$

Figure 6.1: Panels (a,b): Sets of activated nodes ($\mathcal{I}$; in red) and useless nodes ($\mathcal{U}_n$; in orange) at two different times for a diffusion process that starts in the source node set $\mathcal{S} = \{3, 5\}$ relative to a particular sink node (n; in black) . Any path from a useless node to the sink node is *blocked* by an activated node. The set of disabled ($\mathcal{X}_n$) nodes is simply the union of the sets of activated and useless nodes. Panel (c): Sets of disabled nodes $\mathcal{X} \in \Omega_n^*$ such that $\mathcal{S} \subseteq \mathcal{X}$. They represent the states that we need to describe the temporal evolution of a diffusion process towards the sink node n that starts in the set of sources $\mathcal{S}$.

our results can easily be extended to diffusion networks with phase-type pairwise transmission likelihoods. This is important since phase-type distributions can approximate power-laws (Horvath and Telek, 2000), Rayleigh distributions (Asmussen and Nerman, 1996), and also subprobability distributions, which enable us to describe two step traditional generative models, in which with probability $(1 - \beta)$ an activation will never occur, as in Section 3. We refer the reader to Section 2.3 for a definition of phase-type distributions.

6.2.2 Continuous time diffusion process

We consider diffusion and propagation processes that occur over weighted static networks with known (or inferred, using NetRate, from Section 4) connectivity and transmission rates. A diffusion process starts when a source node set $\mathcal{S}$ becomes activated at time $t = 0$ by action of an external source to the network. Then, source nodes try to activate their children (*i.e.*, neighbors that they can reach directly through an outgoing edge). Once a child i gets activated at time t_i, it tries to activate her own children, and so on. For some pairwise transmission likelihoods, it may happen that $t_i \to \infty$ and child i is never activated. Here, we assume that a node i becomes activated as soon as one of her parents (*i.e.*, neighbors that are able to reach node i through an outgoing edge) activates it, and later activations by other parents do not contribute anymore towards the evolution of the diffusion process. As a consequence of this assumption, at any time $t \geq 0$ there may be some nodes and edges in the network that are useless for the spread of the information (be it in the form of a meme, a sales decision or a virus) towards a specific node n. If these nodes get activated and transmit the information to other nodes, this information can only reach n through previously activated nodes. Therefore, the activation time t_n of node n does not depend on these nodes.

By construction, a diffusion process is time-ordered and it induces the structure of a directed acyclic graph (DAG) on the diffusion network (which is *not* acyclic in general). In other words, nodes that are activated later in time cannot infect previously activated nodes.

Finally, given a diffusion process that started in the set of source nodes $\mathcal{S}$, we define $N(\mathcal{S};T)$ as the number of nodes activated up to time T and then define the influence function $\sigma(\mathcal{S};T)$ as the average total number of nodes activated up to time T, *i.e.*, $\sigma(\mathcal{S};T) = \mathbb{E}N(\mathcal{S};T)$.

6.2.3 Continuous time influence maximization problem

Our goal is to find the set of source nodes $\mathcal{S}$ in a diffusion network $(\mathcal{G}, \mathbf{A})$ that maximizes the influence function $\sigma(\mathcal{S};T)$. In other words, the set of source nodes $\mathcal{S}$ such that a diffusion process in $\mathcal{G}$ reaches, on average, the greatest number of nodes before a window cut off T. Thus, we aim to solve:

$$\mathcal{S}^* = \operatorname*{argmax}_{|\mathcal{S}| \leq k} \sigma(\mathcal{S};T), \tag{6.1}$$

where the source set $\mathcal{S}$ is the variable to optimize and the time horizon T and the set cardinality k are constants. We consider a cardinality constraint on $\mathcal{S}$ since in many real-world scenarios, including a node in the set of source nodes entails a cost.

6.3 InfluMax

6.3.1 Influence evaluation

The influence function depends on the probability of activation of every node in the network as follows:

$$\sigma(\mathcal{S};T) = \mathbb{E}N(\mathcal{S};T) = \sum_{n=1}^{N} P(t_n \leq T|\mathcal{S}), \tag{6.2}$$

where t_n is the activation time of node n, $\mathcal{S}$ is the set of source nodes, and T is the time horizon or time window cut-off. Therefore, we need to compute the probability of activation $P(t_n \leq T|\mathcal{S})$ for each node n in the network. Note that whenever $n \in \mathcal{S}$, the probability of activation $P(t_n \leq T|\mathcal{S})$ is trivially 1. We will refer to the node n as sink node.

Revisiting the basic assumptions about a diffusion process that we presented in Section 6.2, we recall some definitions to describe its temporal evolution. Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a set of nodes $\mathcal{B} \subset \mathcal{V}$, and a node $n \in \mathcal{V}$, we define the set of nodes blocked by or dominated by $\mathcal{B}$:

$$\begin{aligned} S_n(\mathcal{B}) =& \{u \in \mathcal{V} : \text{ any path from } u \text{ to } n \text{ in } \mathcal{G} \text{ visits} \\ & \text{at least one node in } \mathcal{B}\}. \end{aligned} \tag{6.3}$$

By definition, $\mathcal{B} \subseteq S_n(\mathcal{B})$ and $S_n(S_n(\mathcal{B})) = S_n(\mathcal{B})$. We now define the set Ω^*_n as:

$$\Omega^*_n = \{\mathcal{X} \subset \mathcal{V} : \mathcal{X} = S_n(\mathcal{X})\}. \tag{6.4}$$

In words, all nodes in $\mathcal{X} \in \Omega^*_n$ block only themselves relative to the sink node n. We can find all sets in Ω^*_n efficiently (Georgiadis et al., 2006; Provan and Shier, 1996). In particular, we are able to find each $\mathcal{X} \in \Omega^*_n$ in time $O(|\mathcal{V}|)$ and although in dense networks $|\Omega^*_n|$ can be large, we are interested in real networks, typically sparse, in which $|\Omega^*_n|$ is significantly smaller.

Given a diffusion process that starts in a set of source nodes $\mathcal{S}$, a sink node n and any time $t \geq 0$, we denote the set of activated nodes as $\mathcal{I}(t|\mathcal{S})$, the set of useless nodes as $\mathcal{U}_n(t|\mathcal{S})$, and the set of *disabled* nodes (*i.e.*, activated or useless) as $\mathcal{X}_n(t|\mathcal{S})$. Useless nodes are nodes that if they get activated and transmit the information to other nodes, this information can only reach the sink node n through previously activated nodes. Figures 6.1(a) and 6.1(b) illustrate the set of activated nodes ($\mathcal{I}$) and the set of useless nodes ($\mathcal{U}_n$) for a diffusion process in a small network at two different times. Note that the set of disabled nodes ($\mathcal{X}_n$) is composed of the sets of activated ($\mathcal{I}$) and useless nodes ($\mathcal{U}_n$). Moreover, by definition of $S_n(\cdot)$, $\mathcal{U}_n(t|\mathcal{S}) = S_n(\mathcal{I}(t|\mathcal{S}))\backslash\mathcal{I}(t|\mathcal{S})$ and $\mathcal{X}_n(t|\mathcal{S}) = S_n(\mathcal{I}(t|\mathcal{S}))$. Now, by studying the temporal evolution of $\mathcal{X}_n(t|\mathcal{S})$ we will be able to compute $P(t_n \leq T|\mathcal{S})$.

First, for a diffusion process that starts in the set of source nodes $\mathcal{S}$, it can be shown that the set of disabled nodes $X_n(t|\mathcal{S})$ at any time $t \geq 0$ belongs to Ω_n^*. The following result is an adaptation of a similar result in (Kulkarni, 1986):

Theorem 10. *Given a set of source nodes $\mathcal{S}$, a sink node n and any time $t \geq 0$, $\mathcal{X}_n(t|\mathcal{S}) \in \Omega_n^*$.*

Proof. Fix $t > t-\varepsilon \geq 0$, where ε is an arbitrarily small time such that no activations occur in $[t-\varepsilon, t)$, and suppose that $\mathcal{X}_n(t-\varepsilon) \in \Omega_n^*$. If no activation occurs at time t, $\mathcal{X}_n(t) = \mathcal{X}_n(t-\varepsilon)$ trivially. If a node $i \in \mathcal{V}$ gets activated at time t, nodes in $S_n(\mathcal{I}(t-\varepsilon|\mathcal{S})\cup\{i\})$ become disabled, $\mathcal{X}_n(t|\mathcal{S}) = S_n(\mathcal{I}(t-\varepsilon|\mathcal{S})\cup\{i\})$, and by definition of $S_n(\cdot)$, $S_n(\mathcal{X}_n(t|\mathcal{S})) = S_n(S_n(\mathcal{I}(t-\varepsilon|\mathcal{S})\cup\{i\})) = S_n(\mathcal{I}(t-\varepsilon|\mathcal{S})\cup\{i\}) = \mathcal{X}_n(t|\mathcal{S})$. In both cases, $\mathcal{X}_n(t-\varepsilon|\mathcal{S}) \in \Omega_n^* \rightarrow \mathcal{X}_n(t|\mathcal{S}) \in \Omega_n^*$. Since $\mathcal{X}_n(0|\mathcal{S}) = S_n(\mathcal{S}) \in \Omega_n^*$, the theorem follows. $\square$

Figure 6.1(c) enumerates all sets of disabled nodes $\mathcal{X} \in \Omega_n^*$ such that $\mathcal{S} \subseteq \mathcal{X}$ for the small network depicted in Figures 6.1(a) and 6.1(b). They represent the states that we need to describe the evolution of a diffusion process that starts in the set of sources $\mathcal{S}$ relative to the sink node n. Now, assuming independent pairwise exponential transmission likelihoods in the diffusion network, we have the following result:

Theorem 11. *((Kulkarni, 1986)) Given a set of source nodes $\mathcal{S}$, a sink node n and independent pairwise exponential transmission likelihoods $f(t_j|t_i;\alpha_{i,j})$, $\{\mathcal{X}_n(t|\mathcal{S}), t \geq$*

$0\}$ *is a continuous time Markov chain (CTMC) with state space* $\{\mathcal{X} : \mathcal{X} \in \Omega_n^*, \mathcal{S} \subseteq \mathcal{X}\}$ *and infinitesimal generator matrix* $Q = [q(\mathcal{D}, \mathcal{B})]$ $(\mathcal{D}, \mathcal{B} \in \{\mathcal{X} : \mathcal{X} \in \Omega_n^*, \mathcal{S} \subseteq \mathcal{X}\})$ *given by:*

$$q(\mathcal{D}, \mathcal{B}) = \begin{cases} \sum_{(i,j)\in C_v(\mathcal{D})} \alpha_{i,j} & \text{if } \exists v \in \bar{\mathcal{D}} : \mathcal{B} = S_n(\mathcal{D} \cup \{v\}), \\ -\sum_{(i,j)\in C(\mathcal{D})} \alpha_{i,j} & \text{if } \mathcal{B} = \mathcal{D}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

where $C(\mathcal{D}) = C(\mathcal{D}, \bar{\mathcal{D}})$ *is the minimal cut between* $\mathcal{D}$ *and* $\bar{\mathcal{D}} = \mathcal{V}\backslash\mathcal{D}$, $C_v(\mathcal{D}) = \{(u, v) \in C(\mathcal{D})\}$, *and* $C(\mathcal{V}) = \emptyset$. *The minimal cut* $C(\mathcal{D})$ *has been shown to be unique (Provan and Ball, 1984).*

Finally, let t_n be the *length* of the fastest (shortest) directed path from any of the nodes in $\mathcal{S}$ to the sink node n in the directed acyclic graph (DAG) induced by the diffusion process on the network $\mathcal{G}$. By construction of the CTMC $\{\mathcal{X}_n(t|\mathcal{S}), t \geq 0\}$ in Theorem 11,

$$t_n = \min\{t \geq 0 : \mathcal{X}_n(t|\mathcal{S}) = S_N | \mathcal{X}_n(0|\mathcal{S}) = S_1\}, \tag{6.6}$$

where S_1 and S_N denote respectively the first and last state of the CTMC. The *length* of the fastest (shortest) path is thus equivalent to the time until the CTMC $\{\mathcal{X}_n(t|\mathcal{S}), t \geq 0\}$ becomes absorbed in the final state S_N starting from state S_1 (*i.e.*, the state in which only the source nodes in $\mathcal{S}$ are activated). Then, computing the probability of activation of the sink node $P(t_n \leq T|\mathcal{S})$ reduces to computing the distribution of time of the sink state of the CTMC. Such distributions are called continuous phase-type distributions. Their generator matrix Q and the cumulative density function satisfy:

$$P(t_n \leq T|\mathcal{S}) = 1 - [1\mathbf{0}]' e^{ST}\mathbf{1}, \text{ where } Q = \begin{bmatrix} S & S^0 \\ \mathbf{0}' & 0 \end{bmatrix}, \tag{6.7}$$

with $S^0 = -S\mathbf{1}$, where $\mathbf{1}$ denotes an all 1's column vector and $\mathbf{0}$ an all 0's column vector. The matrix S results from removing the column and row associated to the last state S_N from Q. By construction, the CTMC $\{X_n(t|\mathcal{S}), t \geq 0\}$ has the structure of a DAG and it is usually sparse. Then, S is upper triangular, sparse and e^{ST} can be computed efficiently (Sidje, 1998).

Our results can be easily extended to diffusion networks with phase-type pairwise transmission likelihoods (Kulkarni, 1986), which can approximate power-laws (Horvath and Telek, 2000) or Rayleigh (Asmussen and Nerman, 1996) transmission likelihoods, which we used previously in Chapter 3 and Chapter 4. Each phase-type pairwise transmission likelihood between a node i and a node j can be viewed as a a CTMC with source node i, sink node j and a generator matrix. By using properties of phase-type distributions (Wolf, 2008), it is possible to show that $\{\mathcal{X}_n(t|\mathcal{S}), t \geq 0\}$ is still a CTMC, with an infinitesimal generator matrix Q that depends on the generator matrices of a set of pairwise transmission likelihoods of the network.

6.3.2 Influence maximization

We have shown how to analytically evaluate our objective function $\sigma(\mathcal{S};T)$ for any set of sources A. However, optimizing $\sigma(\mathcal{S};T)$ with respect to the set of sources $\mathcal{S}$ seems to be a cumbersome task and naive brute-force search over all k node sets is intractable even for relatively small networks. Indeed, we cannot expect to find the optimal solution to the continuous time influence maximization problem defined by Eq. 6.1 since it is NP-hard:

Theorem 12. *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, transmission rates $\mathbf{A} = [\alpha_{j,i}]$, a set of nodes $\mathcal{S} \subseteq \mathcal{V}$ and a time horizon T, the continuous time influence maximization problem defined by Eq. 6.1 is NP-hard.*

Proof. We show that the NP-complete *Set Cover* problem (Karp, 1972) is a special case of our continuous time influence maximization problem. In the Set Cover problem, we are given a finite set $\mathcal{W} = \{w_1, w_2, \ldots, w_n\}$, and a collection of subsets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m \subseteq \mathcal{W}$. Our goal is to know if there exists k of the subsets whose union is equal to $\mathcal{W}$ (We assume that $k < n < m$).

Given an arbitrary instance of the Set Cover problem, we define a corresponding directed bipartite graph with $n + m$ nodes: there is a node i corresponding to each set S_i, a node j corresponding to each element w_j , and a directed edge (i, j) with pairwise transmission likelihood $f(t_j|t_i; \alpha_{i,j}) = \delta(t)$ whenever $w_j \in \mathcal{S}_i$. The Set Cover problem is equivalent to deciding if there is a set $\mathcal{S}$ of k nodes in this graph with $\sigma(\mathcal{S}; T) \geq n + k$. Note that for the instance we have defined, transmission is a deterministic process, as all likelihoods result in paths with probability either 0 or 1 of infecting the ending point of each path before T. Initially activating the k nodes corresponding to sets in a Set Cover solution results in activating all n nodes corresponding to the set $\mathcal{W}$, and if any set $\mathcal{S}$ of k nodes has $\sigma(\mathcal{S}; T) \geq n + k$, then the Set Cover problem must be solvable. □

In absence of source nodes, the expected number of total activations $\sigma(\emptyset, T)$ is zero. By construction, $\sigma(\mathcal{S}; T) \geq 0$. It also follows trivially that $\sigma(\mathcal{S}; T)$ is monotonically nondecreasing in the set of source nodes $\mathcal{S}$, *i.e.*, $\sigma(\mathcal{S}; T) \leq \sigma(\mathcal{S}'; T)$, whenever $\mathcal{S} \subseteq \mathcal{S}'$. Fortunately, we now show that the objective function $\sigma(\mathcal{S}; T)$ is a submodular function in the set of source nodes $\mathcal{S}$. Refer to Section 2.4 for a definition of submodularity. By this natural diminishing returns property, we are able to find a *provable near-optimal* solution to our problem:

Theorem 13. *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, transmission rates $\mathbf{A} = [\alpha_{j,i}]$, a set of nodes $\mathcal{S} \subseteq \mathcal{V}$ and a time horizon T, the influence function $\sigma(\mathcal{S}; T)$ is a submodular function in the set of nodes $\mathcal{S}$.*

Proof. For simplicity, we assume that the activation time of all nodes in $\mathcal{S}$ is $t = 0$; the results generalize trivially. Consider the probability distribution of all possible time differences between each pair of nodes in the network. The activation time of a given node in the network depends only on the time differences drawn from the probability space. Thus, given a sample $\mathbf{\Delta t}$ in the probability space, we define $\sigma_{\mathbf{\Delta t}}(\mathcal{S}; T)$ as the total number of nodes activated in a time less than or equal to T for $\mathbf{\Delta t}$. We will first show that for a given sample $\mathbf{\Delta t}$, $\sigma_{\mathbf{\Delta t}}(\mathcal{S}; T)$ is submodular.

Define $R_{\mathbf{\Delta t}}(k; T)$ as the set of nodes that can be reached from node k in a time shorter than T. We can compute $R_{\mathbf{\Delta t}}(k; T)$ in linear time because it reduces to computing the shortest path between k and the rest of nodes in a DAG. It follows trivially that $\sigma_{\mathbf{\Delta t}}(\mathcal{S}; T) = | \cup_{k \in \mathcal{S}} R_{\mathbf{\Delta t}}(k; T)|$. Define $R_{\mathbf{\Delta t}}(k|\mathcal{N}; T)$ as the set of nodes that can be reached from node k in a time shorter than T and at the same time cannot be reached in a time shorter than T from any node in the set of nodes $\mathcal{N} \subseteq \mathcal{V}$. It follows that $|R_{\boldsymbol{\Delta}\mathbf{t}}(k|\mathcal{N};T)| \geq |R_{\boldsymbol{\Delta}\mathbf{t}}(k|\mathcal{N}';T)|$ for the sets of nodes $\mathcal{N} \subseteq \mathcal{N}'$.

Consider now the sets of nodes $\mathcal{S} \subseteq \mathcal{S}' \subseteq \mathcal{V}$, and a node n such that $n \notin \mathcal{S}'$. Using the definition of submodularity,

$$\begin{aligned}\sigma_{\boldsymbol{\Delta}\mathbf{t}}(\mathcal{S} \cup n;T) - \sigma_{\boldsymbol{\Delta}\mathbf{t}}(\mathcal{S};T) &= |R_{\boldsymbol{\Delta}\mathbf{t}}(n|\mathcal{S};T)| \\ &\geq |R_{\boldsymbol{\Delta}\mathbf{t}}(n|\mathcal{S}';T)| \\ &= \sigma_{\boldsymbol{\Delta}\mathbf{t}}(\mathcal{S}' \cup n;T) - \sigma_{\boldsymbol{\Delta}\mathbf{t}}(\mathcal{S}';T),\end{aligned}$$

and thus $\sigma_{\boldsymbol{\Delta}\mathbf{t}}(\mathcal{S};T)$ is submodular. Now, if we average over the probability space of possible time differences,

$$\sigma(\mathcal{S};T) = \int_{\boldsymbol{\Delta}\mathbf{t}} \sigma_{\boldsymbol{\Delta}\mathbf{t}}(\mathcal{S};T) f(\boldsymbol{\Delta}\mathbf{t})\, d\boldsymbol{\Delta}\mathbf{t} \tag{6.8}$$

is also submodular. □

A well-known approximation algorithm to maximize monotonic submodular functions is the *greedy algorithm*. It adds nodes to the source node set $\mathcal{S}$ sequentially. In step k, it adds the node which maximizes the *marginal gain*

$$n_k = \underset{n \in V \backslash \mathcal{S}_{k-1}}{\operatorname{argmax}}\, \sigma(\mathcal{S}_{k-1} \cup \{n\};T) - \sigma(\mathcal{S}_{k-1};T). \tag{6.9}$$

The greedy algorithm is guaranteed to find a source node set which achieves at least a constant fraction $(1 - 1/e)$ ($\approx 63\%$) of the optimal average total number of activated nodes (Nemhauser et al., 1978).

Moreover, we can also use the submodularity of $\sigma(\mathcal{S};T)$ to acquire a tight *online* bound on the solution quality obtained by *any* algorithm:

Theorem 14 ((Leskovec et al., 2007a)**.** *For a source set $\hat{\mathcal{S}} \subseteq \mathcal{V}$ with k sources and a node $n \in \mathcal{V} \backslash \hat{\mathcal{S}}$, let $\delta_n = \sigma(\hat{\mathcal{S}} \cup \{n\};T) - \sigma(\hat{\mathcal{S}};T)$ and $n_1, \ldots n_k$ be the sequence of k*

nodes with δ_n in decreasing order. Then,

$$\max_{|\mathcal{S}|\leq k} \sigma(\mathcal{S};T) \leq \sigma(\hat{\mathcal{S}};T) + \sum_{i=1}^{k} \delta_{n_i}.$$

We refer to our continuous time influence maximization method as InfluMax.

Speeding-up InfluMax. We can speed up our algorithm by implementing the following speed-ups:

- *Lazy evaluation* (LE, (Leskovec et al., 2007a)): it dramatically reduces the number of evaluations of marginal gains by exploiting the submodularity of $\sigma(\mathcal{S};T)$. Note that lazy evaluation can also be employed to speed-up the computation of the on-line bound for InfluMax.

- *Localized source nodes* (LSN): for each sink node n, we speed up the computation of $P(t_n \leq T|\mathcal{S})$ by ignoring source nodes whose shortest path to the sink node is longer than m hops.

- *Limited transmission paths* (LTP): for each sink node n, we speed up the computation of $P(t_n \leq T|\mathcal{S})$ by ignoring paths longer than m hops from any source in $\mathcal{S}$ to the sink node.

LSN and LTP should be used with care since they provide an approximate evaluation of $P(t_n \leq T|\mathcal{S})$. Therefore, they result on an approximate value of the influence $\sigma(\mathcal{S};T)$. In the remainder of this chapter, if not specified, we run InfluMax with LE but avoid using LSN and LTP.

6.3.3 Experiments on synthetic data

Experimental setup. As in Sections 4.3.2 and 5.3.4, we consider two models of directed real-world networks to generate $\mathcal{G}^*$, namely, the Forest Fire model (Leskovec et al., 2005) and the Kronecker Graphs model (Leskovec et al., 2010). For Kronecker networks, we consider three sets of parameters that produce networks with a very

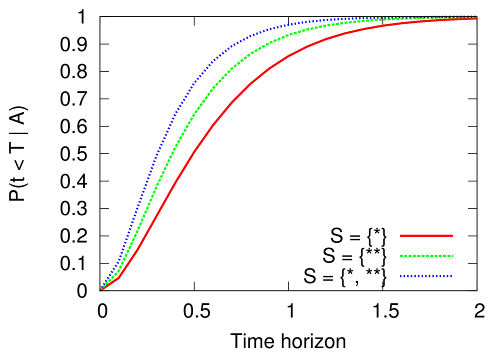
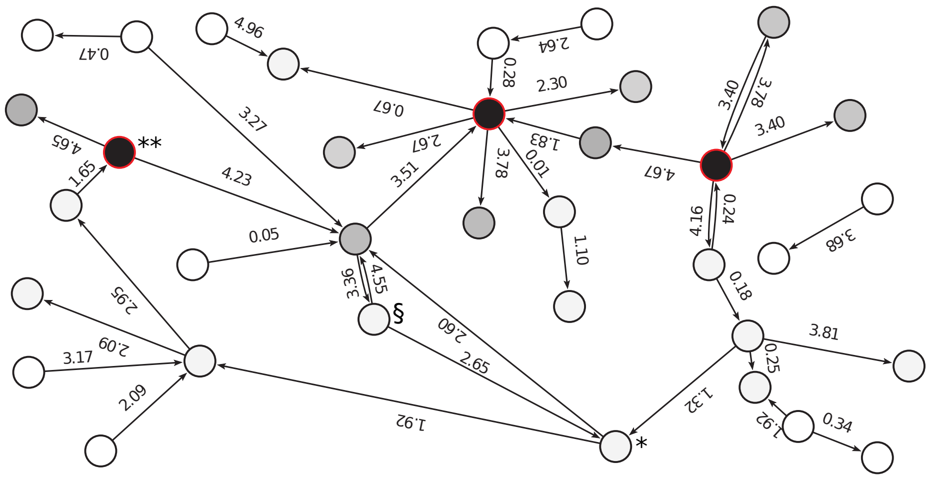


Figure 6.2: Probability of activation vs time horizon. Panel shows the probability of activation (cdf) of node § that results of selecting node $*$, node $**$ or both as source(s) (*i.e.*, $\mathcal{S} = \{*\}$, $\mathcal{S} = \{**\}$, or $\mathcal{S} = \{*, **\}$) on the small core-periphery Kronecker network of Figure 6.3.

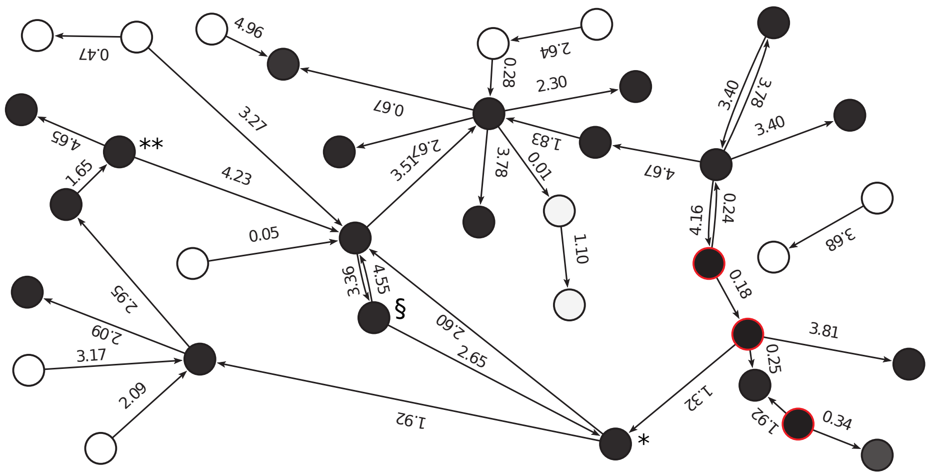
different global network structure: a random network (Erdős and Rényi, 1960), a core-periphery network (Leskovec et al., 2008) and a network with hierarchical community structure (Clauset et al., 2008). A brief introduction to Forest Fire and Kronecker Graph models can be found in Section 2.2.2.

First, we generate a network $\mathcal{G}$ using one of the network models cited above. Then, we draw a transmission rate for each edge $(j, i) \in \mathcal{E}$ from a uniform distribution, a Gaussian distribution or a Rayleigh distribution. We control the transmission rate variance across edges in the network by tuning the parameters values of the distributions. In social networks, transmission rates model how fast information spreads across the network. Given $\mathcal{G}$ and the transmission rates $\alpha_{j,i}$, our aim is to find the most influential subset of k nodes, *i.e.*, the subset of nodes that maximizes the spread of information up to a time T.

Solution quality. We evaluate the solution quality that InfluMax achieves in small and large synthetic networks. First, we run InfluMax on a small network and discuss the source sets that our algorithm selects qualitatively. Second, we compare InfluMax with exhaustive search (enumeration) and several state of the art algorithms on the same small network. By studying a small network in which exhaustive search can be run, we are able estimate exactly how far InfluMax is from the NP-hard to find



(a) $T = 0.1$



(b) $T = 5$

Figure 6.3: Probabilities of activation on a core-periphery Kronecker network with 35 nodes and 39 edges for $T = 0.1, 5$ for the sources selected by the proposed influence maximization algorithm: InfluMax. Nodes with border in red are the optimal influential source nodes. The higher the probability of activation of a node, the darker the color of the node. Pairwise transmission rates α of the pairwise exponential likelihoods are over the edges.

optimum. Finally, we run InfluMax and several state of the art algorithms on different large networks. Running exhaustive search on large networks is computationally too expensive. Therefore, we compute instead the tight on-line bound from Th. 14.

We run InfluMax on a small core-periphery Kronecker network with 35 nodes and 39 edges and look at the probability of activation of every node of the network for the set of sources that InfluMax chooses. We set the transmission rate of every edge of the network by drawing a sample from a uniform distribution $\alpha \sim U(0, 5)$. Figure 6.3 shows the probabilities of activation of each node for two different time horizons. The gray level of each node is proportional to its probability of activation (*i.e.*, white means 0 and black means 1) and the sources have the border in red. We observe that the optimal sources tend to reach sets of nodes with a relatively low overlap. This is a natural consequence of the submodularity of the continuous time influence maximization problem. Moreover, for small time windows, optimal sources are nodes surrounded by high out-going transmission rates while for larger time windows, optimal sources are simply nodes that can reach the greatest amount of nodes. That means, the smaller the time horizon, the more important the temporal dynamics become when choosing the subset of most influential nodes of a given size.

We continue computing the probability of activation of a single node of the network of Figure 6.3 for different source sets and time horizons. Figure 6.3.2 shows the probability of activation (cdf) of the node labeled as § for three different source sets, $\mathcal{S} = \{*\}$, $\mathcal{S} = \{**\}$, and $\mathcal{S} = \{*, **\}$, against the time horizon. Neither the node labeled as $*$ nor the node labeled as $**$ blocks each other with respect to node §. In other words, source $*$ does not block all paths from $**$ to §, and source $**$ does not block all paths from $*$ to §. Therefore, selecting both sources increases the probability of activation of node §, *i.e.*, $P(t_\S \leq T | \{*, **\}) \geq \max(P(t_\S \leq T | \{*\}), P(t_\S \leq T | \{**\}))$. However, if one of the sources blocks the other with respect to a given sink, the probability of activation of the sink given the pair of sources is simply the maximum of the probabilities of activation given each source separately.

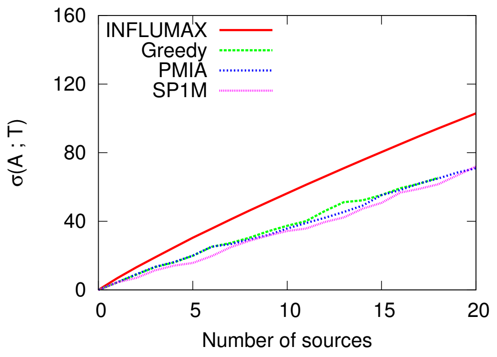
Now, we compare InfluMax with several other state of the art methods. We use the same core-periphery Kronecker network structure of Figure 6.3 and we set the transmission rates by drawing samples from a uniform distribution $\alpha \sim U(0, 10)$. We summarize the results in Table 6.2. We consider several time horizons ($T = 0.1, 0.5, 1.0$) and source set sizes ($|A| = 1, ..., 5$). The first state of the art algorithm is the original greedy algorithm, the second is PMIA and the third is SP1M. The last two methods are essentially efficient heuristics to speed up the greedy algorithm. In addition, we also run a baseline that simply chooses the set of sources randomly. For all methods, we compute the influence they achieve by evaluating Eq. 6.2 for the set of sources selected by them. Surprisingly, InfluMax achieves in most cases the optimal influence that exhaustive search gives but several order of magnitude faster. In other words, the solution given by InfluMax may be in practice much closer to the NP-hard to find optimum than $(1 - 1/e)$, the theoretical guarantee given by Nemhauser et al (Nemhauser et al., 1978). Moreover, InfluMax outperforms all other methods typically by at least 20%.

Finally, we focus on different large synthetic networks. Figure 6.4 shows the average total number of activated nodes against number of sources that InfluMax achieves in comparison with the other methods on a 512 node random Kronecker network, a 1,024 node hierarchical Kronecker network, a 1,024 node core-periphery Kronecker network, and a 1,024 node Forest Fire (scale free) network. All four networks have approximately 2 edges in average per node. We set the time horizon to $T = 1.0$ and the transmission rates are drawn from a uniform distribution $\alpha \sim U(0, 5)$. In all networks, InfluMax typically outperforms other methods by at least 20% by exploiting the temporal dynamics of the network. We also compare InfluMax with the on-line bound from Th. 14. Figure 6.5 shows the average number of activated nodes against number of sources that InfluMax achieves in comparison with the on-line bound for the small core-periphery Kronecker network and the large hierarchical Kronecker network that we used previously. If we pay attention to the value of the bound on the small network for source set sizes significantly smaller than the number of nodes in the network, we observe that the bound value on the influence is not as close to the optimal value given by exhaustive search as we could expect. That means that although the bound is not very tight on the large network, InfluMax may be actually achieving in practice an almost optimal value on that network too.

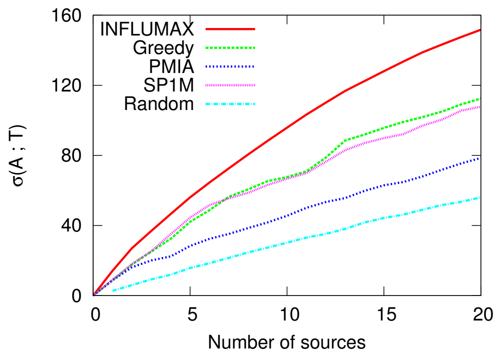
Influence vs. time horizon. Intuitively, the smaller the time horizon, the more important the temporal dynamics become when choosing the subset of most influential nodes of a given size. Figure 6.6 shows the average total number of activated nodes against time horizon for a hierarchical Kronecker network with 1,024 nodes

Algorithm	$\mid\mathbf{A}\mid$	$\sigma(\mathcal{S};\mathbf{0.1})$	$\sigma(\mathcal{S};\mathbf{0.5})$	$\sigma(\mathcal{S};\mathbf{1.0})$
Enumeration	1	3.05	8.95	13.90
	2	5.73	15.24	19.66
	3	8.04	18.01	21.70
	4	9.90	20.10	23.71
	5	11.60	22.17	25.59
InfluMax	1	3.05	8.95	13.90
	2	5.73	15.24	19.66
	3	8.04	18.01	21.70
	4	9.90	20.10	23.71
	5	11.60	22.17	25.59
Greedy (Kempe et al., 2003)	1	3.05	7.70	9.31
	2	4.59	10.67	13.87
	3	6.18	12.70	15.88
	4	7.57	14.84	18.32
	5	8.62	16.05	19.70
PMIA (Chen et al., 2010)	1	1.20	1.67	1.89
	2	2.21	2.84	3.43
	3	4.90	11.67	16.80
	4	5.90	12.31	16.96
	5	8.90	18.03	22.02
SP1M (Chen et al., 2009)	1	2.15	8.04	11.66
	2	3.88	9.75	12.13
	3	4.88	10.75	13.14
	4	6.71	13.47	16.06
	5	7.96	13.95	16.16
Random	1	1.61	3.77	5.34
	2	3.13	6.71	8.99
	3	4.63	9.29	11.84
	4	6.07	11.61	14.35
	5	7.39	13.36	16.04

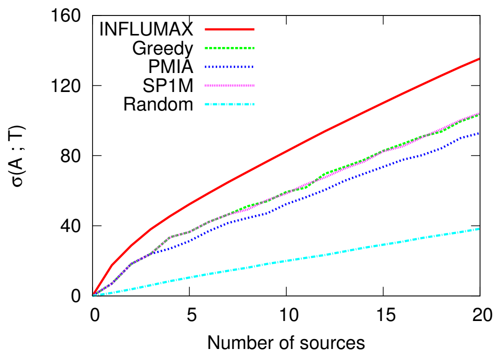
Table 6.2: Influence $\sigma(\mathcal{S};T)$ (*i.e.*, average number of activated nodes) that enumeration (*i.e.*, exhaustive search), InfluMax and several other baselines achieve in the small diffusion network of Fig. 6.3 for different time horizon values T and number of sources $|\mathcal{S}|$. InfluMax always achieves the optimal influence that exhaustive search gives but several order of magnitude faster. Moreover, it typically outperforms other algorithms by more than 20%.



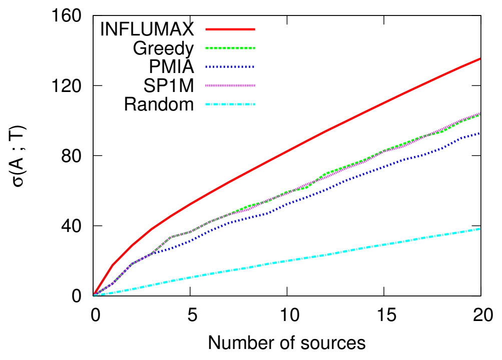
(a) Forest Fire



(b) Random Kronecker



(c) Hierarchical Kronecker

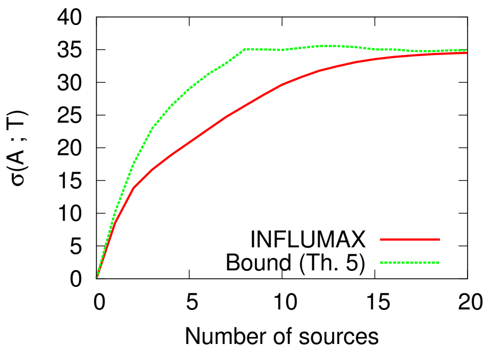


(d) Hierarchical Kronecker

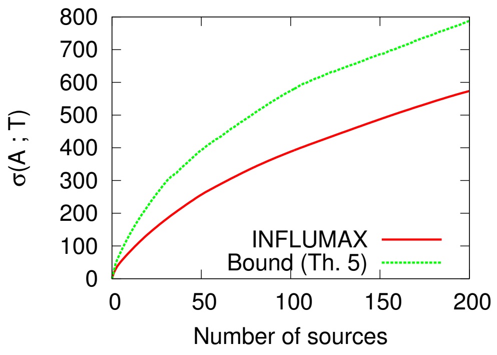
Figure 6.4: Panels plot influence $\sigma(\mathcal{S}; T)$ (*i.e.*, average number of activated nodes) for $T = 1$ and transmission rates drawn from $\alpha \sim U(0, 5)$ against number of sources. (a): 1,024 node Forest Fire network. (b): 512 node random Kronecker network. (c): 1,024 node hierarchical Kronecker network. (d): 1,024 node core-periphery Kronecker network. The proposed algorithm InfluMax outperforms all other methods typically by at least 20%.

and approx. 2 edges per node. We consider a source set of cardinality $|\mathcal{S}| = 10$ and we draw the transmission rate of each edge from a uniform distribution $\alpha \sim U(0, 5)$. The experimental results for all transmission rates configurations confirm the initial intuition, *i.e.*, the difference between InfluMax and other methods is greater for small time horizons.

Influence vs. transmission rates distribution. Up to this point, we have always consider networks in which the transmission rates of the edges are drawn from a



(a) 35 node network



(b) 1,024 node network

Figure 6.5: Influence $\sigma(\mathcal{S};T)$ achieved by InfluMax in comparison with the online upper bound from Theorem 14 for $T = 1$. (a): 35 node network from Figure 6.3. (b): 1,024 node hierarchical Kronecker. InfluMax's performance is close to the upper bound, especially for the smaller network.

uniform distribution. However, in many real networks this is not the case. We now consider the hierarchical network for which we have previously studied the impact of the time horizon on the influence but we set the transmission rates of the edges by drawing samples from (i) a uniform distribution ($\alpha_{min} = 0, \alpha_{max} = 2$), (ii) a Gaussian distribution ($\mu = 1, \sigma = 1$; we rejected any negative samples) and (iii) a Rayleigh distribution ($\sigma = 1$). Figure 6.7 compares InfluMax with the traditional greedy algorithm, PMIA and SP1M for time horizon $T = 1$ and $|\mathcal{S}| = 10$. We observe that the more skewed the transmission rate distribution is, the greater is the difference between InfluMax and other methods.

Scalability. Figure 6.8 shows the average computation time per source added of our algorithm implemented (i) with lazy evaluation, (ii) with lazy evaluation and localized source nodes with $m = 6$ hops and (iii) with lazy evaluation and limited transmission paths with $m = 6$ hops on a single CPU (Mac Book Pro with 2.3 Ghz Dual Core and 4 GB RAM). We use hierarchical Kronecker networks with an increasing number of nodes but approximately the same network density since real networks are usually sparse. Remarkably, the number of hops that we use in localized source nodes and limited transmission paths result in an approximation error for the influence $\sigma(\mathcal{S};T)$

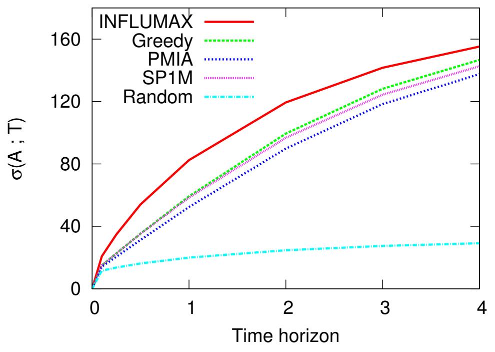


Figure 6.6: Influence $\sigma(\mathcal{S}; T)$ achieved by InfluMax vs. time horizon for a 1,024 node hierarchical Kronecker network. The source set has cardinality $|\mathcal{S}| = 10$ and we draw the transmission rate of each edge from a uniform distribution $\alpha \sim U(0, 5)$. The difference in performance between InfluMax and other methods is greater for small time horizons.

of at most 10%, while achieving an speed-up of $\sim$ 5x for the largest network (2,048 nodes). If desired, we can further trade-off between approximation error and speed. For example, for lazy evaluation and localized source nodes with $m = 5$ on the 2,048 node network, we achieve a speed-up of $\sim$ 9x with a 15% error.

6.3.4 Experiments on real data

Dataset description. We use the original MemeTracker dataset, which contains more than 172 million news articles and blog posts from 1 million online sources over a period of one year from September 1 2008 till August 31 2009[1]. Based on this raw data, we use two different methodologies to trace information on the Web and then create two different datasets: hyperlink cascade dataset and MemeTracker cascade dataset. We refer the reader to Section 2.5.2 for more details on the MemeTracker dataset and an in-depth description of the two methodologies we used to trace information on the Web.

First, from the hyperlink cascade data, we infer an underlying diffusion networks with the top (in terms of hyperlinks) 1,000 media sites and blogs. Second, from the

[1]Data available at `http://memetracker.org` and `http://snap.stanford.edu/netinf`

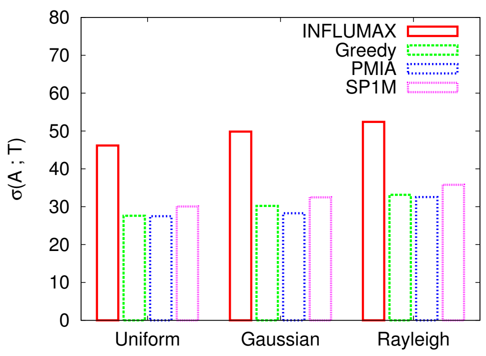


Figure 6.7: Influence $\sigma(\mathcal{S}; T)$ achieved by InfluMax vs. rate distribution for a 1,024 node hierarchical Kronecker network. We draw the transmission rates from three different distributions: Uniform ($\alpha_{min} = 0, \alpha_{max} = 2$), Gaussian ($\mu = 1, \sigma = 1$) and Rayleigh ($\sigma = 1$). The source set has cardinality $|\mathcal{S}| = 10$ and $T = 1$. InfluMax is robust across rate distributions.

MemeTracker cascade data, we infer an underlying diffusion network with the top (in terms of phrases) 1,000 media sites and blogs. As a preprocessing step before running InfluMax on both diffusion networks, we sparsify further the networks by keeping the 1,000 fastest edges since it has been shown that in the context of influence maximization, computations on sparsified models give up little accuracy, but yield significant improvements in terms of scalability (Mathioudakis et al., 2011). Finally, we run InfluMax on the resulting network.

Solution quality. Figure 6.9 shows the average total number of activated nodes against number of sources that InfluMax achieves in comparison with other methods for both real networks, that were inferred from the hyperlink cascade and the MemeTracker cascade datasets, as described above. We set the time horizon to $T = 1.0$. Again, InfluMax outperforms all other methods typically by $\sim 20 - 25\%$, by considering the temporal dynamics of the diffusion.

6.4 Summary

We have developed a method for influence maximization, InfluMax, that accounts for the temporal dynamics underlying diffusion processes. The method allows for

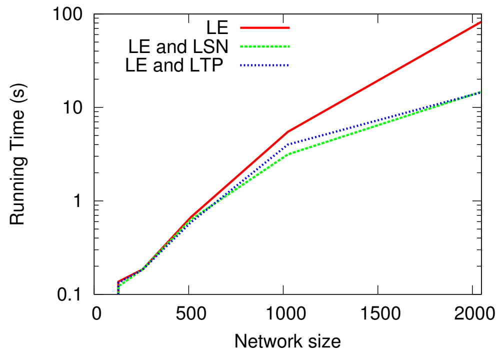


Figure 6.8: Running time. The panel shows the average computation time per source added of InfluMax implemented (i) with lazy evaluation (LE), (ii) with lazy evaluation and localized source nodes (LSN, $m = 6$), and (iii) with lazy evaluation and limited transmission paths (LTP, $m = 6$) against network size (number of nodes). (ii) and (iii) achieve a speed-up of $\sim$ 5x for the largest network (2,048 nodes).

variable transmission (influence) rates between nodes of a network, as found in real-world scenarios. Perhaps surprisingly, for the rather general case of continuous temporal dynamics with variable transmission rates, we can evaluate the influence of any set of source nodes in a network analytically by means of continuous time Markov chains, in contrast to previous approaches that require Monte Carlo simulations (Kempe et al., 2003) or heuristics (Chen et al., 2010, 2009). In this analytical framework, we find the near-optimal set of source nodes that maximizes influence by exploiting the submodularity of our objective function. In addition, the reevaluation of influence for changes on the transmission rates is straightforward and the algorithm parallelizes naturally by sink and source nodes.

We evaluated our algorithm on a wide range of synthetic diffusion networks with heterogeneous temporal dynamics which aim to mimic the structure of real-world social and information networks. Our algorithm is remarkably stable across different network topologies and transmission rates distributions. It outperforms state of the art methods in terms of influence (*i.e.*, average number of activated nodes) for different network topologies, transmission rates distributions, time horizons and source set sizes. InfluMax typically gives an influence gain of $\sim$ 25% and it achieves the greatest improvement for small time horizons; in such scenarios, the temporal

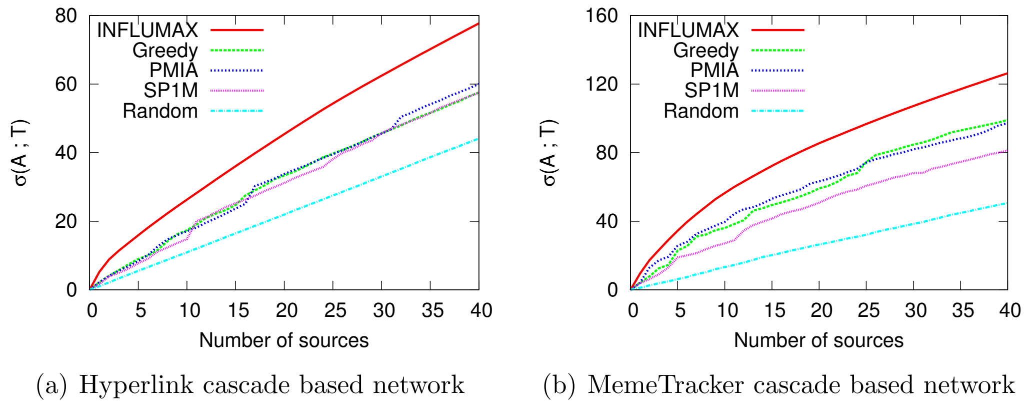


Figure 6.9: Panels plot influence $\sigma(\mathcal{S}; T)$ (*i.e.*, average number of activated nodes) for time horizon $T = 1$ against number of sources for (a) a 1,000 node real diffusion network that we infer from hyperlinks cascades and (b) a 1,000 node real diffusion network that we infer from MemeTracker cascades. The proposed algorithm InfluMax outperforms all other methods by 20-25%. For network inference, we used NetRate (Gomez-Rodriguez et al., 2011).

dynamics play a dramatic role. We also evaluated InfluMax on two real diffusion networks that we inferred from the MemeTracker dataset using NetRate. Again, InfluMax drastically outperformed the state of the art by $\sim 30\%$.

We believe that InfluMax provides a novel view of the influence maximization problem by accounting for the underlying temporal dynamics of diffusion networks.

Chapter 7

Diffusion and survival theory

7.1 Introduction

This chapter presents general theoretical framework to model propagation and infer hidden or unobserved networks using survival theory Gomez-Rodriguez and Schölkopf (2012b). The theoretical framework allows us to generalize previous network inference methods, including NetRate, described in Chapter 4 and consider links which inhibit or encourage the diffusion of a contagion.

In particular, we represent a contagion as a nondecreasing (binary) counting process. We model the instantaneous risk of infection or hazard rate Aalen et al. (2008) of each node using only the infection times of previously infected nodes as covariates. First, we develop an additive risk model under which the hazard rate of each node is additive on the covariates. We show that several previous network inference methods, including NetRate, described in Chapter 4, use particular cases of our more general additive risk model. All these models implicitly consider previously infected nodes to only *increase* the instantaneous risk of infection. Second, we also develop a multiplicative risk model under which the hazard rate of each node is multiplicative on the covariates. This allows previously infected nodes to either *increase* or *decrease* the risk of a node multiplicatively. For example, trendsetters' probability of buying a product may increase when she observes her peers buying a product but may also *decrease* when she realizes that *average, mainstream* friends are buying the product.

Similarly, consider an example of a blog which often mentions pieces of information from a general news media site, but only whenever they are not related to sports. Therefore, if the general news media site publishes a piece of information related to sports, we would like the blog's risk of adopting the information to be smaller than for other type of information. Third, we efficiently fit the parameters of both models using cascade data by the maximum likelihood principle.

The remainder of the chapter is organized as follows: in Section 7.2, we show how to model information propagation over weighted networks using counting processes. Section 7.3 and section 7.4, describe general additive and multiplicative risk models of information propagation. We conclude with a summary of our results in Section 7.5.

7.2 Diffusion as a counting process

Consider a node i, a cascade $\mathbf{t}$, and an indicator function $N_i(t_i)$ such that $N_i(t_i) = 1$ if node i is infected by time t_i in the cascade and $N_i(t_i) = 0$ otherwise. We define the history $\mathcal{F}_{t_i}$ as the set of nodes that has been infected by time t_i and their activation times, *i.e.*, $\mathcal{F}_{t_i} = (\mathbf{t}^{<t_i})$, where $\mathbf{t}^{<t_i} = (t_1 \dots t_N | t_j < t_i)$. By definition, since $N_i(t_i)$ is a nondecreasing counting process, it is a submartingale and satisfies that $E(N_i(t_i)|\mathcal{F}_{t'}) \geq N_i(t')$ for any $t_i > t'$. Then we can decompose $N_i(t_i)$ *uniquely* as $N_i(t_i) = \Lambda_i(t_i) + M_i(t_i)$, where $\Lambda_i(t_i)$ is a nondecreasing predictable process, called *cumulative intensity process* and $M_i(t_i)$ is a mean zero martingale. This is called the Doob-Meyer decomposition of a submartingale (Aalen et al., 2008). Consider $\Lambda_i(t_i)$ to be absolutely continuous and then, there exists a *predictable* intensity process non negative $\lambda_i(t_i)$ such that:

$$N_i(t_i) = \int_0^{t_i} \lambda_i(s)\, ds + M_i(t_i). \tag{7.1}$$

Now, we assume that the intensity process $\lambda_i(t_i)$ depends on a covariate vector $\mathbf{s}(t_i) = \gamma(\mathbf{t}^{<t_i}; t_i)$, where $\gamma(\cdot)$ is an arbitrary time shaping function that we have to decide upon. The covariate vector accounts for the previously infected nodes up to time just before t_i, *i.e.*, the history $\mathcal{F}_{t_i}$. In principle, the covariates can be fixed or time-varying.

Symbol	**Description**
$\mathcal{G}(\mathcal{V},\mathcal{E})$	Directed network with node set V and edge set E
c	Contagion
$\mathbf{t}^c$	Cascade: activation times for contagion c
$\mathcal{C}$	Set of all recorded cascades
t_i^c	Activation time of node i in cascade $\mathbf{t}^c$
T^c	Observation window cut-off or time horizon for cascade $\mathbf{t}^c$
$\mathbf{t}^{\leq T^c}$	Observed activation times for cascade $\mathbf{t}^c c$ up to T^c
$N_i(t_i)$	Indicator function for node i; nondecreasing counting process
$\mathcal{F}_{t_i}$	History for node i up to time t_i
$\lambda_i(t)$	Intensity process for node i
$\mathbf{s}(t_i)$	Covariates for node i at time t_i
$\alpha_i(t_i\|s(t_i))$	Intensity or hazard rate of node i at time t_i
$\boldsymbol{\alpha}_i$	Parameter vector for node i
$\mathbf{A}$	Parameter matrix
$\gamma(\cdot;t_i)$	Time shaping function at time t_i
$f(t_i\|s(t_i))$	Likelihood of activation of node i
$F(t_i s(t_i))$	Probability of activation of node i

Table 7.1: Table of symbols for Chapter 7.

Then, we can rewrite the intensity process of $N_i(t_i)$ as $\lambda_i(t_i) = Y_i(t_i)\alpha_i(t_i|\mathbf{s}(t_i))$, where $Y_i(t_i)$ is an indicator such that $Y_i(t_i) = 1$ if node i is susceptible to be infected just before time t_i and 0 otherwise, and $\alpha_i(t_i|\mathbf{s}(t_i))$ is called intensity or hazard rate of node i and it is defined conditional on the values of the fixed covariates and the observed paths of the time-varying ones. Note that the intensity or hazard rate must be non negative at any time t_i since otherwise $N_i(t_i)$ would decrease, violating the assumptions of our framework. We assume a node is susceptible as long as it did not get infected.

Our goal is to infer a hazard function $\alpha_i(t_i|\mathbf{s})$ for each node i from a set of recorded cascades $\mathbf{C} = \{\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}\}$. This will allow us to evaluate the probability of activation of a susceptible node i at a given time t_i given an arbitrary cascade $\mathbf{t}$ up to time $s < t$ using that

$$F_i(t_i|\mathbf{s}(t_i)) = 1 - e^{-\int_0^{t_i} \alpha_i(t|\mathbf{s}(t))}\, dt, \tag{7.2}$$

which is a well-known result in survival theory. We then compute the likelihood of activation of a susceptible node i at a time t_i, $f_i(t_i|\mathbf{s}(t_i)) = dF_i(t_i|\mathbf{s}(t_i))/dt_i$.

In the remainder of the chapter, we propose an additive and a multiplicative model for the hazard functions $\alpha_i(t_i|\mathbf{s})$ and validate them experimentally in synthetic and real data. There are several reasons to do so. First, to provide a general framework which is flexible enough to fit cascading processes over networks in different domains. Second, to allow for both positive and negative influence of a node in its neighbors' hazard rate, without violating the non negativity of hazard rates over time. Third, it has been argued the necessity of both additive and multiplicative models in traditional survival analysis literature (Aalen et al., 2008).

7.3 Additive risk model of diffusion

Given a cascade $\mathbf{t}$, we propose the hazard function $\alpha_i(t_i|\mathbf{s}(t_i))$ of any node i to be additive on the covariates up to t_i. We then show that this is equivalent to consider an independent cascade model in which a node gets infected once the *first* parent infects her, assumption that has been used extensively in previous work, including our diffusion model for inference of unweighted and weighted networks in chapters 3 and 4.

We consider the hazard rate of any node i in a cascade $\mathbf{t}$ to be:

$$\alpha_i(t_i|\mathbf{s}(t_i)) = \boldsymbol{\alpha}_i^T \mathbf{s}(t_i) = \boldsymbol{\alpha}_i^T \gamma(\mathbf{t}^{<t_i}; t_i), \tag{7.3}$$

where $\boldsymbol{\alpha} \geq \mathbf{0}$ is a positive parameter vector and $\gamma(\cdot) \geq \mathbf{0}$ is an arbitrary positive time shaping function on the previously infected nodes up to t_i. We force the parameter vector and time shaping function to be non negative to avoid ill-defined negative hazard functions at any time t_i. For simplicity, we assume that (a) each covariate depends only on one parent and therefore each parameter only models the effect of a single parent, and (b) we apply the same time shaping function to each of the parents' activation times. That means mathematically that $\gamma(\mathbf{t}^{<t_i}; t_i) = (\gamma(t_1; t_i), \ldots, \gamma(t_{i-1}; t_i))$. Our goal is thus to infer the optimal parameters $\boldsymbol{\alpha}_i$ for every node i that maximize

the likelihood of a set of observed cascades $C\{\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}\}$. To this aim, we need to compute the likelihood of a cascade starting from the hazard rate of each node.

We first compute the cumulative likelihood of activation $F_i(t_i|\mathbf{s}(t_i))$ of a node i from the hazard rate using Eq. 7.2:

$$F_i(t_i|\mathbf{s}(t_i); \boldsymbol{\alpha}_i) = 1 - \prod_{j:t_j<t_i} e^{-\left(\alpha_{j,i} \int_{t_j}^{t_i} \gamma(t_j;t)\right)}. \tag{7.4}$$

Then, the likelihood of activation $f_i(t_i|\mathbf{s}(t_i))$ is:

$$f_i(t_i|\mathbf{s}(t_i); \boldsymbol{\alpha}_i) = \sum_{j:t_j<t_i} \alpha_{j,i}\gamma(t_j;t_i) \prod_{k:t_k<t_i} e^{-\alpha_{k,i} \int_{t_k}^{t_i} \gamma(t_k;t)} \tag{7.5}$$

Now, consider a cascade $\mathbf{t} := (t_1, \ldots, t_N)$. We first compute the likelihood of the observed activations $\mathbf{t}^{\leq T} = (t_1, \ldots, t_N | t_i \leq T)$. Since we assume activations are conditionally independent given the covariates, the likelihood factorizes over nodes as

$$f(\mathbf{t}^{\leq T}; \mathbf{A}) = \prod_{t_i \leq T} f(t_i|\mathbf{s}(t_i); \boldsymbol{\alpha}_i), \tag{7.6}$$

where $\mathbf{A} := \{\alpha_{j,i} \,|\, i,j = 1, \ldots, n, i \neq j\}$. By combining Eq. 7.5 and Eq. 7.6 the likelihood of the activations in a cascade is:

$$f(\mathbf{t}^{\leq T}; \mathbf{A}) = \prod_{i:t_i<T} \sum_{j:t_j<t_i} \alpha_{j,i}\gamma(t_j;t_i) \times \prod_{k:t_k<t_i} e^{-\alpha_{k,i} \int_{t_k}^{t_i} \gamma(t_k;t)\,dt}. \tag{7.7}$$

However, Eq. 7.7 only considers infected nodes. The fact that some nodes are *not* infected during the observation window is also informative. We then add survival terms $(1 - F_n(t_n|\mathbf{s}(t_n); \boldsymbol{\alpha}_n))$ for any node n such that $t_n > T$, and apply logarithms. Therefore, the log-likelihood of a cascade $\mathbf{t}$ is:

$$\begin{aligned}\log f(\mathbf{t}; \mathbf{A}) = \sum_{i:t_i<T} \log\left(\sum_{j:t_j<t_i} \alpha_{j,i}\gamma(t_j;t_i)\right) - \sum_{i:t_i<T} \sum_{k:t_k<t_i} \alpha_{k,i} \int_{t_k}^{t_i} \gamma(t_k;t)\,dt \\ - \sum_{n:t_n>T} \sum_{m:t_m<T} \alpha_{m,n} \int_{t_m}^{T} \gamma(t_m;t)\,dt,\end{aligned} \tag{7.8}$$

Network Inference Method	$\gamma(\mathbf{t_j}; \mathbf{t_i})$
NetRate, InfoPath (Exp)	$I(t_j < t_i)$
NetRate, InfoPath (Pow)	$\max(0, 1/(t_i - t_j))$
NetRate, InfoPath (Ray)	$\max(0, t_i - t_j)$
KernelCascade	$\{k(\tau_l, t_i - t_j)\}_1^m$
moNet	$I(t_j < t_i)\gamma e^{-d(\mathbf{f}_j, \mathbf{f}_i)}$

Table 7.2: Mapping from several network inference methods to our general additive risk model.

where $\mathbf{A} := \{\alpha_{j,i} \,|\, i, j = 1, \ldots, n, i \neq j\}$, the first two terms represents the infected nodes, and the third term represents the surviving ones up to the observation window cut-off T. Perhaps surprisingly, the log-likelihood is jointly concave on the parameters $\alpha_{j,i}$.

Now, assuming independent cascades, the log-likelihood of a set of cascades $C = \{\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}\}$ is the sum of the log-likelihoods of the individual cascades given by Eq. 7.8. Then, we apply the maximum likelihood principle on the log-likelihood of the set of cascades to find the optimal parameters $\boldsymbol{\alpha}_i$ of every node i:

$$\begin{array}{ll} \text{minimize}_{\mathbf{A}} & -\sum_{c \in C} \log f(\mathbf{t}^c; \mathbf{A}) \\ \text{subject to} & \alpha_{j,i} \geq 0,\, i, j = 1, \ldots, N, i \neq j, \end{array} \tag{7.9}$$

where $\mathbf{A} := \{\alpha_{j,i} \,|\, i, j = 1, \ldots, n, i \neq j\}$ are the variables. The solution to Eq. 7.9 is unique and computable:

Theorem 15. *The network inference problem for the additive risk model defined by equation Eq. 7.9 is convex in* $\mathbf{A}$.

Proof. Convexity of Eq. 7.9 follows from linearity, composition rules for convexity, and concavity of the logarithm. □

If $\alpha_{j,i} > 0$, node j increases additively the hazard rate of node i (positive influence). If a parameter $\alpha_{j,i} = 0$, it means node j does not have any influence on the hazard rate of node i – there is not edge between j and i.

We find some common features of the solutions to the network inference problem under the additive risk model. The first term in the log-likelihood of each cascade, defined by Eq. 7.8, ensures infected nodes have at least one parent since otherwise the log-likelihood would be negatively unbounded, *i.e.*, $\log 0 = 1$. Moreover, there exists a natural diminishing property on the number of parents of a node – since the logarithm grows slowly, it weakly rewards infected nodes for having many parents. The second and third term in the log-likelihood of each cascade, defined by Eq. 7.8, consist of positively weighted l1-norm on the vector $\mathbf{A}$. L1-norms are well-known heuristics to encourage sparse solutions (Boyd and Vandenberghe, 2004). That means, optimal networks for the additive risk model are sparse.

7.3.1 Generalizing other network inference methods

The algorithm NetRate, described in chapter 4, and several state of the art network inference algorithms that build on our work (Du et al., 2012; Wang et al., 2012), model information propagation using continuous time generative probabilistic models of diffusion. Such models start by describing the pairwise interactions between pair of nodes. They define a pairwise likelihood $f_i(t_i|t_j; \alpha_{j,i})$ for every node i and j, where $\alpha_{j,i}$ is a transmission rate from node j to node i. Then, they continue computing the likelihood of activation of a node by assuming a node gets infected once the *first* parent infects her, as in the independent cascade model (Kempe et al., 2003). Finally, they conclude by computing the likelihood of a cascade from the likelihoods of activation. Importantly, the following result holds:

Theorem 16. *The independent cascade model in continuous time is an additive hazard model on the pairwise hazards between a node and her parents.*

Proof. In the independent cascade model in continuous time, for a given node i, the likelihood of activation $f_i(t_i|\mathbf{t}^{<t_i}; \mathbf{A})$ and the probability of survival $S_i(t_i|\mathbf{t}^{<t_i}; \mathbf{A})$ given

the previously infected nodes $\mathbf{t}^{<t_i}$ are:

$$\begin{aligned} f(t_i|\mathbf{t}^{<t_i};\mathbf{A}) &= \prod_{k:t_k<t_i} S(t_i|t_k;\alpha_{k,i}) \sum_{j:t_j<t_i} H(t_i|t_j;\alpha_{j,i}), \\ S(t_i|\mathbf{t}^{<t_i};\mathbf{A}) &= \prod_{k:t_k<t_i} S(t_i|t_k;\alpha_{k,i}), \end{aligned}$$

where $H(t_i|t_j;\alpha_{j,i})$ is the pairwise hazard of edge (j,i). Then, the hazard of node i is

$$\alpha_i(t_i|\mathbf{t}^{<t_i};\mathbf{A}) = \sum_{j:t_j<t_i} H(t_i|t_j;\alpha_{j,i}), \tag{7.10}$$

which is trivially additive on the pairwise hazards between a node and her parents. $\square$

Therefore, the additive risk model is a generalization of the independent cascade model in continuous time. The model we used in NetRate (chapter 4) and several models used by other state of the art network inference methods map easily to our general additive risk model, Table 7.2. Pairwise transmission likelihoods used in NetRate result in simple pairwise hazard rates that map into our model by setting the time shaping functions $\gamma(\cdot)$. The *kernelized* hazard functions used in KernelCascade (Du et al., 2012) map into our model by considering m covariates per parent, where $k(\tau_l,\cdot)$ is a kernel function and τ_l is the l^{th} point in a m-point uniform grid over of $[0,T]$. It allows to model multimodal hazard functions. Finally, the featured-enhanced diffusion model used in moNet (Wang et al., 2012) maps into our model by considering a time shaping function with both temporal and non-temporal covariates, where $d(\mathbf{f}_j,\mathbf{f}_i)$ denote the distance between two non-temporal feature vectors and γ is a normalization constant.

7.3.2 Link to Aalen's model

Our additive model is a particular case of the well known Aalen's model in survival theory (Aalen et al., 2008). In Aalen's model, the hazard function of each node i is parametrized as $\alpha_{i,0}(t)+\alpha_i(t)_i^T\mathbf{s}_i(t)$, where $\alpha(t)$ is a vector that accounts for the effect of a collection of observable covariates $\mathbf{s}(t)$ and $\alpha_{i,0}(t)$ is a baseline. In general, Aalen's model allows some individual parameters $\alpha_{j,i}(t)$ to be negative. Unfortunately, in our scenario this would easily lead to ill defined hazards. For example, consider there is only one infected node j before node i gets infected. If $\alpha_{j,i} < 0$, then the hazard rate would be negative, and would result in an ill-defined likelihood of activation for node i. Inspired by Aalen's model, we could however extend our additive model to consider external causes (by adding a baseline $\alpha_{i,0}(t)$), time varying parameters $\alpha_{j,i}(t)$ or other type of covariates $\mathbf{s}_i(t)$, while keeping the parameters non negative.

7.4 Multiplicative risk model of diffusion

Given a cascade $\mathbf{t}$, we propose the hazard function $\alpha_i(t_i|\mathbf{s}(t_i))$ of any node i to be multiplicative on the covariates up to t_i. This allows us to model situations in which a parent can either a increase or decrease the hazard rate of a node, without violating the non negativity of hazard rates any time time t_i. Additive risk models, in contrast, only consider parents to increase the hazard rate of a node.

We consider the hazard rate of any node i in a cascade $\mathbf{t}$ to be:

$$\alpha_i(t_i|\mathbf{s}(t_i)) = \alpha_{0,i}(t_i) \prod_{j:t_j<t_i} \beta_{j,i} \tag{7.11}$$

where $\alpha_{0,i}(t) \geq 0$ is a fixed or time varying baseline, which is independent of the previously infected nodes, endogenous to the observed (monitored) nodes, and $\beta_{j,i} \geq \epsilon > 0$ are the parameters of the model, and represent the *positive* or *negative* influence of node j in node i. If $\beta_{j,i} > 1$, when node j becomes infected, the instantaneous risk of node i increases, if $\beta_{j,i} < 1$, it decreases and, if $\beta_{j,i} = 1$, node j does not have any effect on the risk of node i. The baseline $\alpha_{0,i}(t)$ may be, in general, unknown, and we choose a parametric form based on expert knowledge or additional information.

Our goal is thus to infer the optimal parameters $\beta_{j,i}$ for every node i given a set of cascades $C = \{\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}\}$. To this aim, we need to compute the likelihood of a cascade starting from the hazard rate of each node. We first compute the cumulative likelihood of activation $F_i(t_i|\mathbf{s}(t_i))$ of a node i from the hazard rate using Eq. 7.2:

$$F_i(t_i|\mathbf{s}(t_i);\boldsymbol{\beta_i}) = 1 - \exp\left(-\sum_{j:t_j\le t_i, j>0}\prod_{k:t_k<t_j,k>0}\beta_{k,i}\int_{t_{j-1}}^{t_j}\alpha_{0,i}(t)\,dt\right) \tag{7.12}$$

Then, the likelihood of activation $f_i(t_i|\mathbf{s}(t_i))$ is:

$$f_i(t_i|\mathbf{s}(t_i);\boldsymbol{\beta_i}) = \alpha_{0,i}(t_i)\left(\prod_{k:t_k<t_i}\beta_{k,i}\right)(1 - F_i(t_i|\mathbf{s}(t_i))) \tag{7.13}$$

where we assume the indices to indicate temporal order, $t_0 = 0 < t_1 < \ldots < t_{i-1} < t_i$. The key observation to compute the likelihood of activation from the cumulative likelihood of activation is to realize that there is only one integral in the cumulative likelihood that contains t_i, and the derivative is with respect to this variable t_i.

Now, consider a cascade $\mathbf{t} := (t_1, \ldots, t_N)$. We first compute the likelihood of the observed activations $\mathbf{t}^{\le T} = (t_1, \ldots, t_N | t_i \le T)$. Since we assume activations are conditionally independent given the covariates, the likelihood factorizes over nodes as

$$f(\mathbf{t}^{\le T}; B) = \prod_{t_i\le T} f(t_i|\mathbf{s}(t_i);\boldsymbol{\beta}_i), \tag{7.14}$$

where $B := \{\beta_{j,i}\,|\, i,j = 1,\ldots,n, i \ne j\}$. By combining Eq. 7.13 and Eq. 7.14, the likelihood of the activations in a cascade is:

$$f(\mathbf{t}^{\le T}; B) = \prod_{i:t_i<T}\alpha_{0,i}(t_i)\prod_{k:t_k<t_i}\beta_{k,i}\times\exp\left(\sum_{j:t_j\le t_i}\prod_{k:t_k<t_j,k>0}\beta_{k,i}\int_{t_{j-1}}^{t_j}\alpha_{0,i}(t)\,dt\right). \tag{7.15}$$

However, Eq. 7.15 only considers infected nodes. The fact that some nodes are *not* infected during the observation window is also informative. We then add survival terms $(1 - F_n(t_n|\mathbf{s}(t_n);\boldsymbol{\beta}_n))$ for any node n such that $t_n > T$, use as parameters

$\alpha_{j,i} = \log(\beta_{j,i})$ and apply logarithms to compute the log-likelihood of a cascade as,

$$\begin{aligned}\log f(\mathbf{t}; \mathbf{A}) = &\sum_{i:t_i<T} \sum_{k:t_k<t_i} \alpha_{k,i} + \sum_{i:t_i<T} \log(\alpha_{0,i}(t_i)) \\ &- \sum_{i:t_i<T} \sum_{j:t_j \le t_i} e^{\sum_{k:t_k<t_j, k>0} \alpha_{k,i}} \int_{t_{j-1}}^{t_j} \alpha_{0,i}(t)\, dt \\ &- \sum_{n:t_n>T} \sum_{j:t_j \le T} e^{\sum_{k:t_k<t_j, k>0} \alpha_{k,i}} \int_{t_{j-1}}^{t_j} \alpha_{0,i}(t)\, dt\end{aligned} \tag{7.16}$$

where $\mathbf{A} := \{\alpha_{j,i} \,|\, i, j = 1, \ldots, n, i \neq j\}$, the first three terms represent the infected nodes, and the last term represents the surviving ones up to the observation window cut-off T. The log-likelihood is jointly concave on the parameters $\mathbf{A}$, by exploiting the convexity of e^x and linearity.

Now, assuming independent cascades, the log-likelihood of a set of cascades $C = \{\mathbf{t}^1, \ldots, \mathbf{t}^{|C|}\}$ is the sum of the log-likelihoods of the individual cascades given by Eq. 7.16. Then, if the baselines $\alpha_{0,i}(t)$ are known, we apply the maximum likelihood principle on the log-likelihood of the set of cascades to find the optimal parameters $\boldsymbol{\alpha}_i$ of every node i:

$$\text{minimize}_{\mathbf{A}} \quad -\textstyle\sum_{c \in C} \log f(\mathbf{t}^c; \mathbf{A}) \tag{7.17}$$

where $\mathbf{A} := \{\alpha_{j,i} \,|\, i, j = 1, \ldots, n, i \neq j\}$ are the variables. The solution to Eq. 7.17 is unique and computable:

Theorem 17. *The network inference problem for the multiplicative risk model defined by Eq. 7.17 is convex in* $\mathbf{A}$.

Proof. Convexity of Eq. 7.17 follows from linearity, composition rules for convexity, and convexity of the exponential. □

If $\alpha_{j,i} > 0$, node j increases the hazard rate of node i (positive influence), if $\alpha_{j,i} < 0$, node j decreases the hazard rate of node i (negative influence), and finally if a parameter $\alpha_{j,i} = 0$, node j does not have any influence on the hazard rate of node i – there is not edge between j and i.

However, there are some undesirable properties of the solution to the unconstrained network inference problem for the multiplicative risk model as defined by Eq. 7.17. The optimal network will be dense: any pair of nodes (j, i) that are not infected by the same contagion at least once will have negative influence on each other. Even worse, the negative influence between those pair of nodes will be arbitrarily large, making the optimal solution unbounded.

A first strategy to induce sparsity and obtain a bounded optimal solution consists in allowing only positive influences between nodes, *i.e.*, $\alpha_{j,i} \geq 0$ for any (j, i), and solving instead:

$$\begin{array}{ll} \text{minimize}_{\mathbf{A}} & -\sum_{c \in C} \log f(\mathbf{t}^c; \mathbf{A}) \\ \text{subject to} & \alpha_{j,i} \geq 0,\, i, j = 1, \ldots, N, i \neq j, \end{array} \tag{7.18}$$

where $\mathbf{A} := \{\alpha_{j,i} \,|\, i, j = 1, \ldots, n, i \neq j\}$ are the variables. In this way, the first term in the log-likelihood of each cascade, defined by Eq. 7.16, becomes a l1-norm on the vector $\mathbf{A}$, and we encourage sparse networks (Boyd and Vandenberghe, 2004). Moreover, the optimal solution is not unbounded anymore.

However, whenever we work in a domain in which negative influence between nodes may occur, we propose a second strategy. First, if a pair (j, i) does not get infected in any common cascades, we set $\alpha_{j,i}$ to zero and do not include $\alpha_{j,i}$ in the log likelihood. This rules out interactions between nodes that got infected in disjoint set of cascades and avoid unbounded optimal solutions. In other words, we assume that if node j has a (positive or negative) influence in node i, they should get infected at least once by the same contagion at times $t_j < t_i < T$. Second, although by ruling out interactions between nodes that got infected in disjoint set of cascades we also reduce the network density of the optimal solution, the solution is not encouraged to be sparse yet. We then achieve greater sparsity by including a l1-norm regularization parameter in $\mathbf{A}$ (Boyd and Vandenberghe, 2004). Therefore, we finally solve:

$$\text{minimize}_{\mathbf{A}} \quad -\sum_{c \in C} g(\mathbf{t}^c; \mathbf{A}) + \lambda \sum_j \sum_i |\alpha_{j,i}|, \tag{7.19}$$

where λ is a sparsity penalty factor, and $g(\mathbf{t}^c; \mathbf{A})$ is the log-likelihood of cascade $\mathbf{t}^c$ which omits parameters $\alpha_{j,i}$ of pairs (j, i) that did not get infected at least once in the same cascade at times $t_j < t_i < T$. The problem is still convex by applying the same reasoning as in Th. 17.

7.4.1 Link to Cox's model

Our multiplicative model is a particular case of the well known Cox's model in survival theory (Aalen et al., 2008). In Cox's model, the hazard function of each node i is parametrized as $\lambda_{i,0}(t)e^{\alpha_i^T \mathbf{s}(t)}$, where α is a vector that accounts for the effect of a collection of observable covariates $\mathbf{s}(t)$ and $\lambda_{i,0}(t)$ is a baseline. Future work could use Cox's model to consider unknown non parametric baselines $\lambda_{i,0}(t)$ by fitting the model using partial likelihood, Cox's goodness of fit tests to validate further the model or other type of covariates $\mathbf{s}(t)$.

7.5 Summary

In this chapter, we have proposed a theoretical framework to model information propagation over implicit or unobserved networks using survival theory. First, we proposed an additive risk model under which the network inference problem from cascades can be solved efficiently by exploiting convexity. We have shown that several state of the art network inference methods use particular instances of our additive risk model. Then, we proposed a multiplicative risk model under which the network inference problem can be solved efficiently too by exploiting convexity. This multiplicative model allows for situations in which a parent can either encourage or inhibit the diffusion of a contagion.

Chapter 8

Conclusions and future work

This dissertation has been mainly devoted to one of the fundamental research problems in the context of network diffusion, inference of hidden or implicit networks over which various types of *contagions* spread. Additionally, we have also tackled the influence (spread) maximization over networks from a novel perspective. In particular, we have made the following contributions:

- **Network inference.** We introduced probabilistic models and network inference algorithms for modeling diffusion over hidden or implicit networks. We first developed two algorithms, NetInf and Multitree, which allow to infer the structure of unweighted static diffusion networks, where diffusion occurs at the same rate across different edges. The algorithms exploit submodularity to efficiently find suboptimal networks with *provable* guarantees using cascade data. The algorithms require to set the number of edges of the inferred network. We then developed NetRate, an algorithm which allow to infer the structure and rates of weighted static and dynamic diffusion networks in which diffusion occurs at different rates across different edges from cascade data. The algorithm exploits convexity to efficiently find optimal networks using cascade data. It naturally (without heuristics) imposes sparse solutions and requires no parameter tuning. We applied our inference algorithms to information diffusion among mainstream media and blogs sites and experiment with more than

340 million blogs and news articles. We find that diffusion networks tends to have a core-periphery structure with a small set of core media sites that diffuse information to the rest of the Web. These sites tend to have stable circles of influence with more general news media sites acting as connectors between them. Information pathways for general recurrent topics are more stable across time than for on-going news events. Clusters of news media sites and blogs often emerge and vanish in matter of days for on-going news events. Major social movements and events involving civil population, such as the Libyan's civil war or Syria's uprise, lead to an increased amount of information pathways among blogs as well as in the overall increase in the network centrality of blogs and social media sites.

- **Influence maximization.** We developed InfluMax, an algorithm for influence maximization which uses an asynchronous continuous time model of diffusion that takes into account heterogeneous temporal dynamics. In other words, diffusion can occur at different rates across different edges, as in real-world examples. We showed experimentally on synthetic and real diffusion networks that InfluMax outperforms other state of the art algorithms by considering the heterogeneous temporal dynamics of diffusion.

We believe that these contributions open several interesting avenues for future research. We envision the following short term goals:

- **Threshold model in continuous time.** We have developed continuous time models of diffusion which are inspired on the sequential independent cascade model – we assume a node gets activated due to the *first parent*. However, the sequential threshold cascade model has been also argued for in the literature (Kempe et al., 2003). It would be interesting to develop continuous time models of diffusion based on the sequential threshold cascade model.

- **Beyond temporal features.** Here we only used time difference to infer edges and thus it would be interesting to utilize more informative features (*e.g.*, textual content of postings, credibility, etc.) to more accurately estimate the transmission or cascade likelihoods.

- **External sources and events detection.** We notice that many times the changes in the inferred dynamic networks could be attributed to sudden external real-world events. How can diffusion network inference be combined with methods for detecting external influence in networks (Myers et al., 2012)? And also, how can dynamic network inference be extended for detecting unexpected real-world events based on a stream of documents?

- **Signed networks.** Many times, not only information but also sentiment attached to a piece of information spreads through the network (Miller et al., 2011). It would be interesting to think about inference of signed networks, where a positive/negative valence of an edge models sentiment relationship between a pair of nodes.

- **Competing cascades.** In our work, we have assumed contagions to propagate independently. However, this is over simplistic, as noticed recently (Prakash et al., 2012b; Myers and Leskovec, 2012). It would be interesting to relax this assumption in our framework.

- **Scalable influence maximization with temporal dynamics.** Evaluating influence exactly, as InfluMax does, becomes prohibitive in terms of computational complexity when the network density grows. Therefore, it would be interesting to find approximate ways to evaluate influence.

In the context of online media, such modifications and extensions would allow us to improve our understanding of the current landscape of news coverage, the role that news media plays in framing the discussion of important topics, and the evolving ecosystem that news media occupies. Among the medium and long term goals, we also envision:

- **Other applications for network inference.** There are many other domains where our network inference algorithms or some variations of them may be useful: inferring interaction networks in systems biology (protein-protein and gene interaction networks), neuroscience (inferring physical connections between neurons) and epidemiology.

- **Sequential vs. asynchronous diffusion models.** Finding culprits or end effectors from diffusion traces, reconstructing incomplete diffusion traces, spread maximization and minimization are all examples of research problems where diffusion of information, influence or behavior must be modeled. In previous work, diffusion has typically been modeled as a sequential synchronous process in which time is discretized in epochs and propagation is synchronous. Such models are often assumed to facilitate theoretical analysis, including NP-hardness results, but unfortunately they lack realism. Instead, in our work, we provide a novel more realistic view of diffusion as discrete networks of continuous asynchronous temporal processes occurring at different rates. Perhaps surprisingly, our model allows for theoretical analysis as well as fitting real observed data, as shown for the network inference problem and the influence maximization problem. We believe it can be fruitfully applied to other lines of research, including the ones mentioned above.

Appendix A

Table of symbols

Symbol	Description
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Directed network with node set $\mathcal{V}$ and edge set $\mathcal{E}$
$\mathbf{A}$	Pairwise transmission rates for all pair of nodes (i, j)
$(\mathcal{G}, \mathbf{A})$	Diffusion network: directed network $\mathcal{G}$ and transmission rates $\mathbf{A}$
$\alpha_{i,j}$	Pairwise transmission rate of edge (i, j)
c	Contagion
$\mathbf{t}^c$	Cascade: activation times for contagion c
$\mathcal{C}$	Set of all recorded cascades
t_i^c	Activation time of node i in cascade $\mathbf{t}^c$
T^c	Observation window cut-off or time horizon for cascade $\mathbf{t}^c$
$f(t_j \| t_i, \alpha_{i,j})$	Pairwise transmission likelihood of edge (i, j)
$F(t_j \| t_i, \alpha_{i,j})$	Cumulative density function of edge (i, j)
$S(t_j \| t_i; \alpha_{i,j})$	*Survival function* of edge (i, j)
$H(t_j \| t_i; \alpha_{i,j})$	*Hazard function*, or instantaneous activation rate, of edge (i, j)
$\mathcal{E}_\varepsilon$	Set of ε-edges, $\mathcal{E} \cap \mathcal{E}_\varepsilon = \emptyset$ and $\mathcal{E} \cup \mathcal{E}_\varepsilon = \mathcal{V} \times \mathcal{V}$
$\mathcal{T}_c(\mathcal{G})$	Set of all possible propagation trees of cascade $\mathbf{t}^c$ on network $\mathcal{G}$
$\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$	Cascade propagation tree, $\mathcal{T} \in \mathcal{T}_c(\mathcal{G})$
$\mathcal{V}_\mathcal{T}$	Node set of T, $\mathcal{V}_\mathcal{T} = \{i \mid i \in \mathcal{V}\text{and } \mathbf{t}_c[i] < \infty\}$
$\mathcal{E}_\mathcal{T}$	Edge set of T, $\mathcal{E}_\mathcal{T} \subseteq \mathcal{E} \cup \mathcal{E}_\varepsilon$
$\mathcal{C}_t$	Set of recorded cascades by time t
$\mathbf{t}^{\leq T^c}$	Observed activation times for cascade $\mathbf{t}^c c$ up to T^c

Continued on next page...

Symbol	**Description**
$g(\mathbf{A})$	Prior likelihood on the transmission rates $\mathbf{A}$
$\mathcal{A}$	Support of the prior likelihood on the transmission rates $g(\mathbf{A})$
T	Time horizon
$\mathcal{S}$	Node source set
s_i	Source node i
$N(\mathcal{S};T)$	Number of activated nodes at time t given a node source set $\mathcal{S}$
$\sigma(\mathcal{S};T) = \mathbb{E}N(\mathcal{S};T)$	Influence function
n	Sink node
$S_n(\mathcal{B})$	Set of nodes dominated by B with respect to a sink node n
Ω_n^*	Set of self dominant node sets with respect to a sink node n
$\mathcal{I}(t\|\mathcal{S})$	Activated node set at time t given $\mathcal{S}$
$\mathcal{U}_n(t\|\mathcal{S})$	Useless node set at time t given $\mathcal{S}$ and n
$\mathcal{X}_n(t\|\mathcal{S})$	Set of disable nodes at time t given $\mathcal{S}$ and n
$N_i(t_i)$	Indicator function for node i; nondecreasing counting process
$\mathcal{F}_{t_i}$	History for node i up to time t_i
$\lambda_i(t)$	Intensity process for node i
$\mathbf{s}(t_i)$	Covariates for node i at time t_i
$\alpha_i(t_i\|s(t_i))$	Intensity or hazard rate of node i at time t_i
$\boldsymbol{\alpha}_i$	Parameter vector for node i
$\gamma(\cdot;t_i)$	Time shaping function at time t_i
$f(t_i\|s(t_i))$	Likelihood of activation of node i
$F(t_i s(t_i))$	Probability of activation of node i

Table A.1: Table of symbols.

References

O.O. Aalen, Ø. Borgan, and H.K. Gjessing. *Survival and event history analysis: a process point of view.* Springer Verlag, 2008.

E. Adar and L. A. Adamic. Tracking information epidemics in blogspace. In *Web Intelligence*, pages 207–214, 2005.

A. Agarwal and J.C. Duchi. Distributed delayed stochastic optimization. In *NIPS '11: Advances in Neural Information Processing*, 2011.

A. Ahmed and E. Xing. Recovering time-varying networks of dependencies in social and biological studies. In *PNAS '09: Proceedings of the National Academy of Sciences*, volume 106, 2009.

R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

S. Aral and D. Walker. Identifying Influential and Susceptible Members of Social Networks. *Science*, 337(6092):337–341, 2012.

S. Asmussen. *Applied probability and queues.* Springer, 2003.

S. Asmussen and O. Nerman. Fitting phase-type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, 23(4):419–441, 1996.

F. Bach, E. Moulines, et al. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *NIPS '11: Advances in Neural Information Processing*, 2011.

L. Backstrom and J. Leskovec. Supervised random walks: Predicting and recommending links in social networks. In *WSDM '11: Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, 2011a.

L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In *WSDM '11: Proceedings of the 4th ACM international conference on Web search and data mining*, 2011b.

N. T. J. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications.* Hafner Press, 2nd edition, 1975.

A.-L. Barabási. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207, 2005.

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

S. Bharathi, D. Kempe, and M. Salek. Competitive influence maximization in social networks. *Internet and Network Economics*, pages 306–311, 2007.

F. Bießmann, J.-M. Papaioannou, M. Braun, and A. Harth. Canonical trends: Detecting trend setters in web data. *NIPS '12: Advances in Neural Information Processing*, 2012.

D. Blatt, A.O. Hero, and H. Gauchman. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1):29–51, 2008.

A. Blum, H. Chan, and M. Rwebangira. A random-surfer web-graph model. *ANALCO 06: Proceedings of the 3rd Workshop on Analytic Algorithmics and Combinatorics*, 2006.

L. Blume, D. Easley, J. Kleinberg, R. Kleinberg, and É. Tardos. Which networks are least susceptible to cascading failures? In *FOCS '11: Proceedings of the 52nd IEEE Annual Symposium on Foundations of Computer Science*, pages 393–402, 2011.

S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

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

C. Budak, D. Agrawal, and A. El Abbadi. Limiting the Spread of Misinformation in Social Networks. In *WSDM '11: Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, 2011.

A.J. Butte and I.S. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Pac Symp Biocomput*, volume 5, pages 418–429, 2000.

T. Carnes, C. Nagarajan, S.M. Wild, and A. van Zuylen. Maximizing influence in a competitive social network: a follower's perspective. In *EC '07: Proceedings of the 9th international conference on Electronic commerce*, pages 351–360. ACM, 2007.

W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD '10: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038. ACM, 2010.

W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincon, X. Sun, Y. Wang, W. Wei, and Y. Yuan. Influence maximization in social networks when negative opinions may emerge and propagate. In *SDM '11: Proceedings of the SIAM Conference on Data Mining*, pages 379–390, 2011.

F. Chierichetti, J. Kleinberg, and D. Liben-Nowell. Reconstructing Patterns of Information Diffusion from Incomplete Observations. In *NIPS '11: Advances in Neural Information Processing Systems*, 2011.

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

P. Domingos and M. Richardson. Mining the network value of customers. In *KDD '01: Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001.

N. Du, L. Song, A. Smola, and M. Yuan. Learning networks of heterogeneous influence. In *NIPS '12: Advances in Neural Information Processing*, 2012.

J.C. Duchi, A. Agarwal, M. Johansson, and M.I. Jordan. Ergodic subgradient descent. In *Allerton '11: Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing*, 2011.

J. Edmonds. Optimum branchings. *Journal of Resesearch of the National Bureau of Standards*, (71B):233–240, 1967.

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

M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 251–262. ACM, 1999.

A. D. Flaxman, A. M. Frieze, and J. Vera. A geometric preferential attachment model of networks. *Internet Mathematics*, 3(2):187–205, 2006.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostat*, 9(3):432–441, 2008.

N. Friedman and D. Koller. Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1): 95–125, 2003.

N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive datasets: The "Sparse Candidate" algorithm. In *UAI '99: Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1999.

N. Fyson, T. De Bie, and N. Cristianini. The netcover algorithm for the reconstruction of causal networks. *Neurocomputing*, 2012.

L. Georgiadis, R.F. Werneck, R.E. Tarjan, S. Triantafyllis, and D.I. August. Finding dominators in practice. *Journal of Graph Algorithms and Applications*, 10(1):69–94, 2006.

L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *The Journal of Machine Learning Research*, 3:707, 2003.

Z. Ghahramani. Learning dynamic Bayesian networks. *Adaptive Processing of Sequences and Data Structures*, 1998.

R. Ghosh and K. Lerman. A framework for quantitative analysis of cascades on networks. In *WSDM '11: Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pages 665–674, 2011.

M. Gomez-Rodriguez and B. Schölkopf. Influence Maximization in Continuous Time Diffusion Networks. In *ICML '12: Proceedings of the 29th International Conference on Machine Learning*, 2012a.

M. Gomez-Rodriguez and B. Schölkopf. Modeling Information Propagation with Survival Theory. In *NIPS '12: Advances in Neural Information Processing Systems. Workshop in Algorithmic and Statistical Approaches for Large Social Networks*, 2012b.

M. Gomez-Rodriguez and B. Schölkopf. Submodular Inference of Diffusion Networks from Multiple Trees. In *ICML '12: Proceedings of the 29th International Conference on Machine Learning*, 2012c.

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

M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the Temporal Dynamics of Diffusion Networks. In *ICML '11: Proceedings of the 28th International Conference on Machine Learning*, 2011.

M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring Networks of Diffusion and Influence. In *ACM Transactions on Knowledge Discovery from Data (TKDD)*, volume 5, 2012.

M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf. Structure and Dynamics of Information Pathways in On-line Media. In *WSDM '13: Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, 2013.

A. Goyal, F. Bonchi, and L.V.S. Lakshmanan. Learning influence probabilities in social networks. In *WSDM '10: Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 241–250. ACM, 2010a.

A. Goyal, F. Bonchi, L.V.S. Lakshmanan, M.F. Balcan, N.J.A. Harvey, R. Lapus, F. Simon, P. Tittmann, S. Ben-Shimon, A. Ferber, et al. Approximation Analysis of Influence Spread in Social Networks. *Arxiv preprint arXiv:1008.2005*, 2010b.

M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. `http://cvxr.com/cvx`, 2010.

D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pages 491–501, 2004.

A. Horvath and M. Telek. Approximating heavy tailed behaviour with phase type distributions. 2000.

L. Hufnagel, D. Brockmann, and T. Geisel. Forecast and control of epidemics in a globalized world. *PNAS '04: Proceedings of the National Academy of Sciences*, 101 (42):15124, 2004.

Y. Ijiri and H. Simon. Some distributions associated with bose-einstein statistics. *PNAS '75: Proceedings of the National Academy of Sciences*, 72(5):1654–1657, 1975.

M.T. Irfan and L.E. Ortiz. A game-theoretic approach to influence in networks. In *AAAI '11: Proceedings of the 25th Conference on Artificial Intelligence*, 2011.

R. Jansen, H. Yu, D. Greenbaum, Y. Kluger, N.J. Krogan, S. Chung, A. Emili, M. Snyder, J.F. Greeblatt, and M. Gerstein. A bayesian networks approach for predicting proteinprotein interactions from genomic data. *Science*, 302(5644):449–453, 2003.

E.H. Kaplan. What are the risks of risky sex? Modeling the AIDS epidemic. *Operations Research*, 37(2):198–209, 1989.

R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 1972.

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

D. Kempe, J. M. Kleinberg, and É. Tardos. 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*, 2003.

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

J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: Measurements, models, and methods. *Computing and Combinatorics*, pages 1–17, 1999.

V.G. Kulkarni. Shortest paths in networks with exponentially distributed arc lengths. *Networks*, 16(3):255–274, 1986.

R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *FOCS '00: 41st Annual Symposium on Foundations of Computer Science*, pages 57–65, 2000.

R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. Structure and evolution of blogspace. *CACM '04: Communications of the ACM*, 47(12):35–39, 2004.

T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila. Finding effectors in social networks. In *KDD '10: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1059–1068, 2010.

J.F. Lawless. *Statistical models and methods for lifetime data*. Wiley New York, 1982.

J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD '05: Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, page 187, 2005.

J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, 2006a.

J. Leskovec, A. Singh, and J. M. Kleinberg. Patterns of influence in a recommendation network. In *PAKDD '06: Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 380–389, 2006b.

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

J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst. Cascading behavior in large blog graphs. In *SDM '07: Proceedings of the SIAM Conference on Data Mining*, 2007b.

J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web*, 2008.

J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking 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.

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

D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM '03: Proceedings of the 12th ACM International Conference on Information and. Knowledge Management*, pages 556–559, 2003.

David Liben-Nowell and Jon Kleinberg. Tracing the flow of information on a global scale using Internet chain-letter data. *PNAS '08: Proceedings of the National Academy of Sciences*, 105(12):4633–4638, 2008.

C. Lippert, O. Stegle, Z. Ghahramani, and K.M. Borgwardt. A kernel method for unsupervised structured network inference. In *AISTATS '09: Proceedings of the Artificial Intelligence and Statistics*, 2009.

M. Lipsitch, T. Cohen, B. Cooper, J.M. Robins, S. Ma, L. James, G. Gopalakrishna, S.K. Chew, C.C. Tan, M.H. Samore, et al. Transmission dynamics and control of severe acute respiratory syndrome. *Science*, 300(5627):1966, 2003.

R. Dean Malmgren, Daniel B. Stouffer, Adilson E. Motter, and L. A. N. Amaral. A poissonian explanation for heavy tails in e-mail communication. *PNAS '08: Proceedings of the National Academy of Sciences*, 105(47):18153–18158, 2008.

M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In *KDD '11: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2011.

N. Meinshausen and P. Buehlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, pages 1436–1462, 2006.

M. Miller, C. Sathi, D. Wiesenthal, J. Leskovec, and C. Potts. Sentiment flow through hyperlink networks. In *ICWSM '11: Proc. of the AAAI International Conference on Weblogs and Social Media*, 2011.

S. Myers and J. Leskovec. On the Convexity of Latent Social Network Inference. In *NIPS '10: Advances in Neural Information Processing Systems*, 2010.

S. Myers, J. Leskovec, and C. Zhu. Information Diffusion and External Influence in Networks. In *KDD '12: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012.

S.A. Myers and J. Leskovec. Clash of the contagions: Cooperation and competition in information diffusion. In *ICDM '12: Proc. of the 12th IEEE International Conference on Data Mining*, 2012.

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

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574, 2009.

P. Netrapalli and S. Sanghavi. Finding the graph of epidemic cascades. In *Proceedings of the 12th ACM SIGMETRICS/Performance*, 2012.

W. K. Newey and D. L. McFadden. Large sample estimation and hypothesis testing. In R. F. Engle and D. L. McFadden, editors, *Handbook of Econometrics, Volume IV*, pages 2111–2245. Elsevier Science B.V., 1994.

D. Pennock, G. Flake, S. Lawrence, E. Glover, and C. Giles. Winners don't take all: Characterizing the competition for links on the web. *PNAS '02: Proceedings of the National Academy of Sciences*, 99(8):5207–5211, 2002.

B. Prakash, J. Vreeken, and C. Faloutsos. Spotting culprits in epidemics: How many and which ones? In *ICDM '12: Proceedings of the 12th IEEE International Conference on Data Mining*, pages 11–20, 2012a.

B.A. Prakash, A. Beutel, R. Rosenfeld, and C. Faloutsos. Winner takes all: competing viruses or ideas on fair-play networks. In *WWW '12: Proceedings of the 21st International Conference on World Wide Web*, pages 1037–1046, 2012b.

J.S. Provan and M.O. Ball. Computing network reliability in time polynomial in the number of cuts. *Operations Research*, 32(3):516–526, 1984.

J.S. Provan and D.R. Shier. A paradigm for listing (s, t)-cuts in graphs. *Algorithmica*, 15(4):351–372, 1996.

W. Reed and M. Jorgensen. The double pareto-lognormal distributiona new parametric model for size distributions. *Communications in Statistics-Theory and Methods*, 33(8):1733–1753, 2004.

M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, page 70, 2002.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

E. M. Rogers. *Diffusion of Innovations.* Free Press, New York, fourth edition, 1995.

D. M. Romero, B. Meeder, and J. Kleinberg. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In *WWW '11: Proceedings of the 20th International Conference on World Wide Web*, 2011a.

D.M. Romero, B. Meeder, and J. Kleinberg. Differences in the mechanics of information diffusion across topics: Idioms, political hashtags, and complex contagion on twitter. In *WWW '11: Proceedings of the 20th international conference on World wide web*, pages 695–704. ACM, 2011b.

N.L. Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for strongly-convex optimization with finite training sets. 2012.

S. Sadikov, M. Medina, J. Leskovec, and H. Garcia-Molina. Correcting for missing data in information cascades. In *WSDM '11: Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, 2011.

K. Saito, M. Kimura, K. Ohara, and H. Motoda. Learning continuous-time information diffusion model for social behavioral data analysis. *Advances in Machine Learning*, pages 322–337, 2009.

K. Saito, K. Ohara, Y. Yamagishi, M. Kimura, and H. Motoda. Learning diffusion probability based on node attributes in social networks. *Foundations of Intelligent Systems*, pages 153–162, 2011.

M. Schmidt, A. Niculescu-Mizil, and K. Murphy. Learning graphical model structure using l1-regularization paths. In *AAAI '07: Proceedings of the 21th Conference on Artificial Intelligence*, volume 22, 2007.

B. Schroeder, S. Damouras, and P. Gill. Understanding latent sector errors and how to protect against them. *8th USENIX FAST*, 2010.

H. Seal. Survival probabilities based on pareto claim distributions. *Astin Bulletin*, 11(1):61–72, 1980.

D. Shah and T. Zaman. Rumors in a network: Who's the culprit? *IEEE Transactions on Information Theory*, 57(8):5163–5181, 2011.

R.B. Sidje. Expokit: a software package for computing matrix exponentials. *ACM Transactions on Mathematical Software*, 24(1):130–156, 1998.

SNAP. SNAP: Stanford Network Analysis Platform. `http://snap.stanford.edu`. 2012.

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

L. Song, M. Kolar, and E. Xing. Time-varying dynamic bayesian networks. In *NIPS '09: Advances in Neural Information Processing Systems*, 2009.

B. Taskar, M. F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *NIPS '03: Advances in Neural Information Processing Systems*, 2003.

InfoPath. InfoPath: `http://snap.stanford.edu/infopath/`. 2013.

NetRate. NetRate: `http://www.stanford.edu/~manuelgr/netrate/`. 2011.

G. Ver Steeg, R. Ghosh, and K. Lerman. What stops social epidemics? In *ICWSM '11: Proceedings of the 5th Int. Conf. on Weblogs and Social Media*, 2011.

J. Vert and Y. Yamanishi. Supervised graph inference. In *Advances in Neural Information Processing Systems*, 2005.

M. J. Wainwright, P. Ravikumar, and J. D. Lafferty. High-dimensional graphical model selection using l1-regularized logistic regression. In *PNAS '06: Proceedings of the National Academy of Sciences*, 2006.

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

C. Wang, J.C. Knight, and M.C. Elder. On computer viral infection and the effect of immunization. In *ACSAC '00: 16th Annual Conference on Computer Security Applications*, pages 246–256, 2000.

L. Wang, S. Ermon, and J. Hopcroft. Feature-enhanced probabilistic models for diffusion network inference. In *ECML PKDD'12: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2012.

D. Watts and S. Strogatz. The small world problem. *Collective Dynamics of Small-World Networks*, 393:440–442, 1998.

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

B. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, 1988.

J. Winick and S. Jamin. Inet-3.0: Internet topology generator. Technical report, Technical Report CSE-TR-456-02, University of Michigan, 2002.

V. Wolf. *Equivalences on Phase-Type Processes.* PhD thesis, University of Mannheim, 2008.

J. Yang and J. Leskovec. Modeling information diffusion in implicit networks. In *ICDM '10: Proceedings of the IEEE International Conference on Data Mining*, 2010.

J. Yang and J. Leskovec. Patterns of Temporal Variation in Online Media. In *WSDM '11: Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, 2011.