CSE 8803 EPI: Data Science for Epidemiology, Fall 2020

Lecturer: B. Aditya Prakash
Scribe: Tanya Churaman

Lecture #8
September 21, 2020

---

1 Summary of Lecture Content

This lecture is meant to give a sense of what sort of inference problems arise and the techniques employed to attempt to solve them. Some of these problems are very challenging. Some are still considered unsolved with no best solution how to solve them. The lecture first introduces how having more data can allow for inference of a network, or at least properties about the network. We focus on the Cascade Transmission model which can help us infer the possible ways the disease could have spread. However, there are other models that can be used to determine the best number of cascades under different assumptions.

We briefly learn about Model Based Reasoning – a different approach from the Statistical Machine Learning Approach. It utilizes various models to represent the data and then recalibrations once given more real-time data. The best model is determined from this set.

Lastly, we focus on Reverse Engineering Epidemics. In many cases, we know what has happened concerning an epidemic. Using this information, we try to work backwards and determine how the epidemic began. We talk about various approaches of finding the most likely source within a network; however, these approaches are very difficult – falling under the difficulty of NP-Complete, NP-Hard, and even #P Hard.

2 Calibration vs Inference

In previous lectures, we focused on how to construct networks from first principles, collect data, and how mobility is an important component. In addition, many of these models are stochastic, which then develop raw mobility data to be analyzed.

This lecture, however, focuses on approaches that involve statistical machine learning. Using a given dataset, we can infer how to model these networks and perhaps, gather other supplementary information to help develop implications.

These approaches are used when we have a lot of data. Previously, we have been studying infection rates; however, it is very difficult to gather data about all the infection propagation patterns for the entire population. However, in other domains, such as social/web cascades, there is a lot of data present. While there is the problem of the network being unknown, the data can be used to infer the propagation network.

In order to utilize this data, Machine Learning methods can be employed, using surveillance information as input. An example of surveillance information is a time-series of infections. This being said, data can also be more complicated – in the form of cascading data.

3 Network Inference Problem

The problem focuses on how to utilize data and ML methods to infer the network.

3.1 Examples of Network Inference

There are many examples of this problem:

- The nodes and edges are unknown or partially unknown
- Graph models with unknown parameters
- MLE (Maximum Likelihood Estimation) Formulation to determine the mostly likely graph given the data
- Analyzing how much data is enough to create the original network
  - Getting data may not be easy to accomplish. Plus, the model should be robust.

3.2 Inferring Netorks Using Traces

In *Figure 1* [5], there is the true network and three cascades c_1, c_2, c_3. The true network can represent propagation of a disease spreading through a population. The true network, however, is unknown. The given data is the cascades which can be used to infer the true network.

c_1 shows how a mean spread from one node to another. c_2 and c_3 show how the mean was spread in different ways. These cascades separately represent how a certain medium is spread within the network.

In an epidemiology scenario, a node may not be getting infected with the same disease repeatedly, or there are multiple diseases occurring within the population that needs to be tracked. Therefore, cascades can be used to represent these different variables and can be emulated to relay further information about the true network.

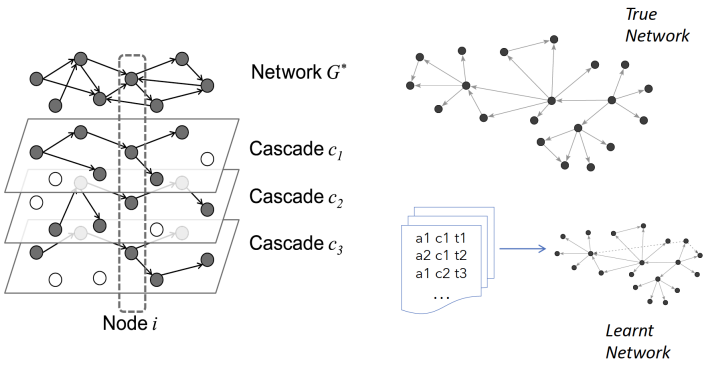

Figure 1: Network from Gomez-Rodriguez + SIGKDD 2010 [5]

3.3 Formulating the Problem

In this problem we are given the cascade set $C = c_1, c_2, ..c_m$. C is the set that consists of cascades 1 to m. We want to utilize C in order to infer the static hidden directed propagation $G*$. Going back to *Figure 1*, we want to use c_1, c_2, c_3 in order to infer $G*$.

We can represent each cascade by its attributes:

- u_i - node id/name
- t_i - time at which cascade c reached node u_i
- ϕ_i - set of features, either from node or contagion

Looking back at Figure 1, we can consider a_1 to be a node that cascade c_1 reached at time t_1. Potentially, auxiliary information can be associated with the node or cascade. For example, think of the node a_1 as a person. A person comes with the supplemental demographic information, such a gender, age, etc. The cascade c_1 could come with supplemental information, such as how deadly the disease it and other factors that could impact its contagion. This auxiliary information can help reveal trends.

Within this problem only $t's$ – the hit times – are observed and $\phi's$ are considered constant. Thus, we should think of t as what time each person/node u_i is affected by the cascade – in terms of epidemiology, when that node got infected.

We can think of each cascade c as a series of time stamps: $c = [t_1, t_2, t_n]$, where n is the number of nodes in $G*$. Within each cascade are the times each node is affected by the cascade. However, $t_u = \infty$ if a node u is not reached during time t. In Figure 1, at Cascade c_1, two white nodes are not affected at that time. Thus, their value would be set to ∞.

3.4 Cascade Transmission Model

From formulating the problem, the Cascade Transmission Probabilistic Model was created. This model is represented by $P(c(u, v))$, which represents the probability that a cascade c propagated from node u to v.

Within this model, here are the following assumptions:

- Every node v can be influenced by only one node u – in reality that might not be true (e.g social media); however, for simplicity's sake, this is a fair assumption.
- Influence structure of c will be a directed tree T – node u will have the infection first and then infect node v, thus resulting in a directed edge from u to v.
- $P(c|T)$ represents the probability cascade c propagated in pattern T.
- $P(c|G)$ represents the probability cascade c occurs in graph G.
- $P(C|G)$ represents the probability a set of cascades C occur in graph G.

Using these probabilities, we can create the MLE for $G*$:

$$G* = \arg\max P(C|G)$$

The goal is to estimate $G*$ in such that we maximize the probabilities of all the cascades c occurring. In Figure 1, we see that $G*$ is an estimated network that tries to incorporate cascades c_1, c_2, c_3.

Let $Tc(G)$ represent the set of all directed spanning trees induced by graph G induced by the nodes infected in cascade c. We are looking at every possible spanning tree to understand how the infection spread in all of the nodes affected in cascade c. We would able to determine the source of the contagion spread and the most probable propagation pattern.

As a result, we can rewrite the former probabilities as:

- $P(c|T) = \prod_{(i,j)\in T(G)} P_c(i,j)$
  - This is just multiplying the edge based probability for each edge in the tree.
- $P(c|G) = \prod_{T\in T(G)} P(c|T)P(T|G) \propto \sum_{T\in T(G)} \prod_{(i,j)\in T} P_c(i,j)$
  - The probability of seeing a cascade c in a graph G is the probability of seeing a cascade c in a tree T times the probability of seeing the tree T in graph G summed over all the possible trees T in G. We want to account for all possible trees, not just a single tree T. $P(T|G)$ is a constant because all trees T are equally likely. Therefore, we convert that probability to the summation of $\sum_{T\in T(G)}$. Thus, we now have $\prod_{T\in T(G)} P(c|T)$. $P(c|T)$ can be substituted with $P_c(i,j)$.
- $P(C|G) = \prod_{c\in C} P(c|G)$
  - Each cascade is assumed to be independent, thus, $P(C|G)$ is just the product the probability $P(c|G)$ for each cascade c.
- Through substitution, the original problem $G* = \arg\max_{|G|\leq k} P(C|G)$ becomes:
  - $P(C|G) = \prod_{c\in C} P(c|G) = \prod_{c\in C} \sum_{T\in T(G)} P(c|T)$

The above final formula for $G* = \arg\max_{|G|\leq k} P(C|G)$ is not easy to compute. However, rather than considering all trees T, we can only consider the most likely propagation tree T per cascade c. Thus, we can rewrite $P(C|G)$ as:

$$P(C|G) = \prod_{maxT\in T(G)} P(c|T)$$

This version is easier for the likelihood function. Below is the new log-likelihood function over empty graph $(\overline{K})$

$$F_c(G) = max_{T\in T(G)} log(P(c|T)) - max_{T\in \overline{K}} log(P(c|T))$$

Therefore, we have resulted in:

$$\textit{Objective Function: } F_c(G) = \sum_{c \in C} F_c(G)$$

$$\textit{Optimal Network: } G* = argmax_{|G| \leq k} F_c(G)$$

In order to understand how a disease could have spread (when this information is unknown), we can look at all the possible ways this could have occurred. With generative simulations, we can get all the potential occurrences how the the infection spread. However, when there is limited data, we have to infer the likely way this disease spread. Therefore, we take the summation of all the possible ways this disease could have spread within G. The inference is not a guarantee, but it is a high likelihood possibility.

4 Other Extensions

The above problem of trying to infer the network can be extended into many more complicated problems.

- Dynamic Networks [4]
  - Right now, we are assuming the networks and cascades are static; however, in reality, mostly everything is dynamic.
- Different Cascade Models [1]
  - Currently, we assume that the cascades follow the IC Model where each edge has a probability of spreading the infection; however other models exist.
- Theoretical analysis of how many samples do we need – in order to derive the graph robustly. [1, 2]. (See Section 4.1)
- More accurate/efficient/robust algorithms [3]
  - Account for noise, faster algorithms, and more accuracy (MLE is an assumption).

4.1 Trace/Cascade Complexity

This problem consists of trying to determine the number of cascades needed in order to infer properties of the network/graph. How much data is needed to infer the graph robustly? There have been some results for this problem that include:

- $\Omega(\frac{n\Delta}{log^2\Delta})$ traces are necessary and $O(n\Delta log(n))$ traces are sufficient to infer the edge set of a graph in the SIR model [1]
  - This formula is a fair number of cascades to infer the edge set of an SIR Model. This claim is unexpected because it implies that the number of cascades needed is not quadratic to the number of nodes in the graph. Roughly, only $nlogn$ traces are needed which is much more efficient that a quadratic amount of traces.

- Exact inference of trees using $O(log(n))$ traces – every node has only one parent node, thus reducing the number of traces!
- Infer the degree distribution using $O(n)$ traces
  - We may not be able to derive the graph from the data; however, we can derive some sort of data about the graph. Thus, getting the inferred degree distribution can allows us to further create the inferred graph using said distribution.
- Similar results under other assumptions [7]

4.2 Inference Problems for Disease Parameters

In Homework 1, we did simple calculations for the parameter estimates of a given network for the SIR Model. However, there are more complicated methods to infer the parameters.

In general, we are given a network G and a set of infections I. We want to estimate $\beta*$ such that the probability that the set I to be infected in G is maximized.

$$\beta* = \arg\max P(I \text{ is infected in } G|\beta)$$

But, how much do we trust I and G? If we trust I and G, then the problem is more well defined. However, G and/or I may be too noisy or have stochastic problems in certain areas. Furthermore, other given parameters and G might have uncertainty too.

Then, there is the question of "Where did the disease start?", which would require reverse engineering. Reverse engineering involves us knowing what has happened and then using that information to piece together what might have happened at the beginning of the scenario. Think of COVID-19. Once it became an issue, people tried to identify the source of what happened. We discuss this in more detail in Section 6.

5 Model Based Reasoning

Given a dataset, we calibrate it to many models, and then, use them to emulate the epidemic.

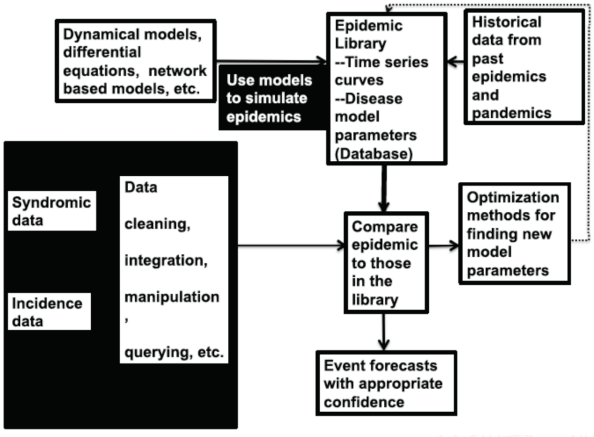

Figure 2: This diagram depicts the process of model based reasoning.

Once given the real dataset, we compare the models and re-calibrate to re-estimate the parameters. We then determine which model represents the real data better and review what assumptions were made to model the epidemic most accurately. *Figure 2* shows this process. This is a different approach from the statistical MLE approach. The model based approach is better with a limited dataset since it prevents the inference of a robust model. Thus, rather than trying to create a model, we try reverse engineer of what could have happened and pick the best model based on the real data.

6 Reverse Engineering Epidemics

The premise of reverse engineering is finding the culprits of the epidemic. When we observe a set of people with a disease, we want to reverse engineer by establishing the contacts between these people to find patient 0 – where the disease started. Sometimes, solving this problem is very difficult. In a midst of many infected, how do we pinpoint the source? Finding the source of the epidemic is important. Once the source is pinpointed, health officials can take the necessary steps to learn about and potentially try to control the outbreak.

These problems are probability based. We can only say that it is highly probable this is how the epidemic began since it is difficult to find the "ground truth" of what truly happened.

6.1 Finding Sources (Culprits)

Figure 3 represents the Culprit's Problem. We have a 2-D grid in which a node is connected to all of its neighbors. The dark grey nodes are the infected nodes. The goal is to determine the order that these nodes were infected. The best place to start is to determine who started the spread of the disease. A way to think of this problem is like a wine spill. When wine is spilled on the carpet, it spreads homogeneously from the center and then outwards, making a circular blotch. We can use this line of thinking to this problem –the first person who caused the epidemic is somewhere in the middle. In the figure, our predicted guess is the blue dots, and the actual culprit is the red dots. Our guess was very close. Of course, this problem can be more complicated. There are other ways to formulate this problem, such as risk minimizing, MLE, MDL, etc.

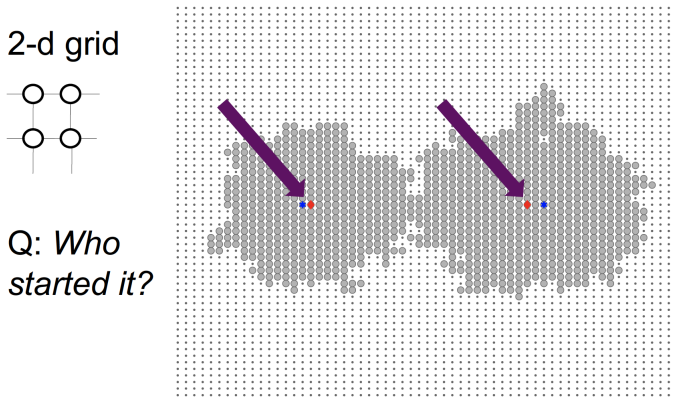

Figure 3: The 2-D grid to depict the Culprit Problem

6.2 One Formulation: Risk Minimization Approach

This problem can be formulated as an optimization problem [6]. The input is a single snapshot of the network and the activation state of nodes. The activation state can be referred to whether or not the node is infected. We want to determine who could be deemed as a good set of initiators – which individuals serve as the root cause of the epidemic.

In *Figure 4*, there is the input data. We want to find out which of the infected individuals started the spread of the disease. In order to determine the potential root causes, a cost function is needed to model to determine the probability that the set of individuals {A,B} started the epidemic. We then optimize that cost function by finding the set that maximizes it. Looking at the figure, we see that a potential final state is the two individuals A and B. The cost function test all possible pairs of infected people to maximize the cost.

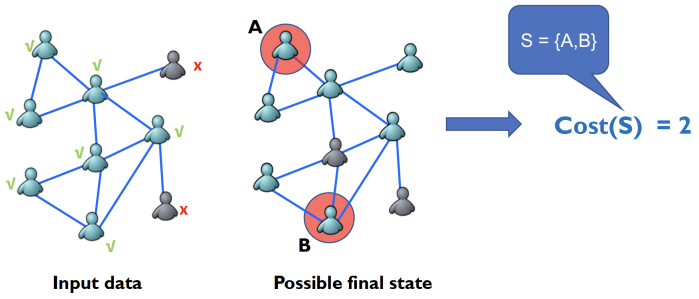

Figure 4: The Good Effectors Approach. The green check mark people are the infected while the red x's are not infected

6.2.1 The k-Effectors Problem

In *Figure 4*, we see the cost function $Cost(S)$. This function can be determined by the difference between the observed activations and the expected activations given S [6].

However, there is a new problem – understanding the probability of how many people that set {A,B} will infect. This is the k-effectors problem – as defined below:

> k-Effectors Problem: Given a social network graph $G = (V, E, p)$ and an activation vector $\mathbf{a}$, find a set X of active nodes (effectors), of cardinality at most k, such that $C(X) = \sum_{v \in V} |\mathbf{a}(v) - \alpha(v, X)|$ is minimized.

The cost $C(X)$ is defined as: for every node v, we subtract the activation state (infected or not infected) and the expected state of that node. The goal is to determine if someone infects a set of nodes X, what is the probability that everyone in the network will get infected as well. We want the cost to be minimized so we minimize the activation and expected states.

6.2.2 Complexity of the k-Effectors Problem

The k-Effector problem in arbitrary graphs is NP-complete. It unlikely to have an optimal, scalable algorithm for all graphs, but it is probable to approximate the algorithm. Unfortunately, the k-Effector problem in arbitrary graphs is NP-Hard to approximate, meaning

a robust, scalable algorithm that is nearly optimal is unlikely. However, the k-Effector problem can be solved optimally in polynomial time on trees. If we assume the graph is a tree, then this problem can be solved efficiently. In a a tree, a node only has one parent — thus, there is only one way in which the disease could have been spread.

6.3 Alternative Formulation: MLE Approach

An alternative method to find the initial source of the epidemic [8] is to use an MLE function. The k-Effectors Problem tries to determine k best sources of the epidemic while the MLE alternative tries to pinpoint the single best source of the epidemic.

$$\hat{v} = \arg\max_{v \in G_N} P(G_N | v^* = v)$$

The formula above is the MLE function. This function is finding the vertex v in G_N – the infected subgraph – in such that probability of seeing the infected graph is maximized. Let v^* be the starting point. We want to set v^* to a vertex v in order to maximize the probability of a graph G_N with many infected nodes. In order to fully grasp this problem, we need to understand the concept of Propagation Ripples.

6.3.1 Propagation Ripples

The first step is to define what a ripple is. *Figure 5* can be used to try to define this term.

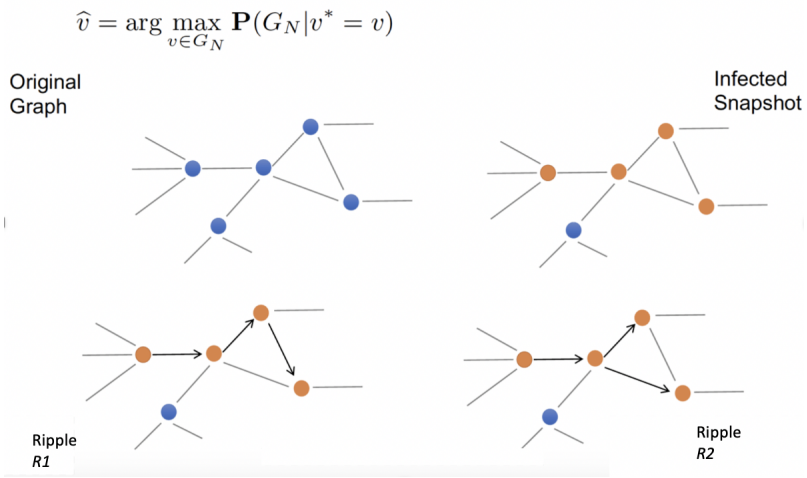

Figure 5: Propagation Ripples

On the left, is the original graph with 100% blue nodes. On the right, is the infected snapshot. All of the orange nodes are infected, and the singular blue node is not infected. What is the possible way that this disease spread that could result in the infected snapshot?

At $R1$, we see a potential path ("ripple") of how the disease spread through the network. Starting at the leftmost orange node, we can follow the directed network to see the infection propagation. $R1$ is not the only way the disease could have rippled through the network. $R2$ is a different path of how the disease spread. Nevertheless, for both $R1$ and $R2$, the

infected snapshot is achieved. Other ripples can be determined as well.

When determining the most likely source, we have to understand the potential ripples that could have occurred in the network. Then, we can identify potential source nodes.

6.3.2 Rumor Centrality

The probability of each ripple occurring is different. Therefore, when calculating the probability of the most likely source node, we must take into account the probability of the ripples. We want to find the node with the most likely set of ripples. When calculating the probability of the ripples, we must account for both the infected and non-infected nodes because non-infected nodes also have the probability of being infected.

By using ripples, the probability of the infected graph being maximized by the maximum number of ripples can be represented. Rumor centrality can be used to rewrite the formula from earlier as:

$$\hat{v} = \arg\max_{v \in G_N} P(G_N | v^* = v)$$

$$P(G_N | v^* = v) = \sum_{\text{all ripples}} R_i$$

The most likely vertex v is the one with the highest rumor centrality.

We take the sum of the ripples because they are mutually exclusive events. Unfortunately, calculating the total number of ripples is a #P-Hard problem. However, the number of ripples can be computed efficiently with a non-trivial algorithm for k-regular trees. While this problem is not as easy as k-effectors, the intuition to solving it is similar.

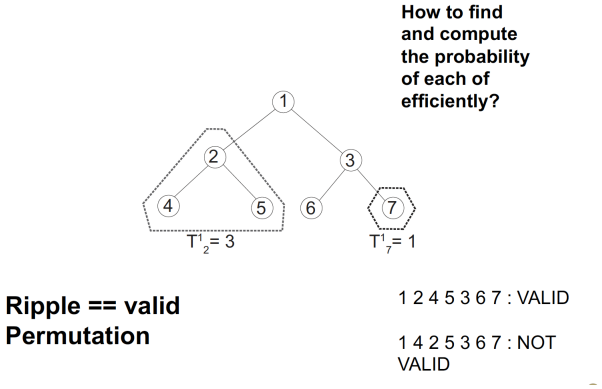

Figure 6: Computing Rumor Centrality Using k-Regular Trees. In this 2-regular tree, each node is expected to have 2 children nodes

In *Figure 6*, the tree is a k-regular tree. We can see that a ripple is a valid permutation on this graph in which the disease can be spread. Whichever node i has the maximum number of permutations has an increased probability of being the source node of the epidemic.

References

[1] Bruno Abrahao, Flavio Chierichetti, Robert Kleinberg, and Alessandro Panconesi. 2013. Trace Complexity of Network Inference. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. Association for Computing Machinery, New York, NY, USA, 491–499. https://doi.org/10.1145/2487575.2487664

[2] Hadi Daneshmand, Manuel Gomez-Rodriguez, Le Song, and Bernhard Schölkopf. 2014. Estimating Diffusion Network Structures: Recovery Conditions, Sample Complexity & Soft-thresholding Algorithm. *CoRR* abs/1405.2936 (2014). arXiv:1405.2936 http://arxiv.org/abs/1405.2936

[3] Nan Du, Le Song, Ming Yuan, and Alex J. Smola. 2012. Learning Networks of Heterogeneous Influence. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2780–2788. http://papers.nips.cc/paper/4582-learning-networks-of-heterogeneous-influence.pdf

[4] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. 2011. Uncovering the Temporal Dynamics of Diffusion Networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11)*. Omnipress, Madison, WI, USA, 561–568.

[5] Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. 2012. Inferring Networks of Diffusion and Influence. *ACM Trans. Knowl. Discov. Data* 5, 4, Article 21 (Feb. 2012), 37 pages. https://doi.org/10.1145/2086737.2086741

[6] Theodoros Lappas, Evimaria Terzi, Dimitrios Gunopulos, and Heikki Mannila. 2010. Finding Effectors in Social Networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*. Association for Computing Machinery, New York, NY, USA, 1059–1068. https://doi.org/10.1145/1835804.1835937

[7] Praneeth Netrapalli and Sujay Sanghavi. 2012. Learning the Graph of Epidemic Cascades. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '12)*. Association for Computing Machinery, New York, NY, USA, 211–222. https://doi.org/10.1145/2254756.2254783

[8] Devavrat Shah and Tauhid Zaman. 2011. Rumors in a Network: Who's the Culprit? *Information Theory, IEEE Transactions on* 57 (09 2011), 5163 – 5181. https://doi.org/10.1109/TIT.2011.2158885