# On the Sample Complexity of Learning Social Influence Functions: A Survey

Alejandro Gomez-Leos, Shashank Gupta

March 2023

## 1 Introduction

Consider the following problem. Suppose we are tasked to engineer a successful product marketing campaign with a limited budget. Amongst many strategies, we consider that of *celebrity marketing*, in which we first convince some initial product adopters—who have high social status—to endorse our product. Then, the hope is that their followers eventually decide to adopt the product (by means of their social platforms), as well as the followers-of-the-followers, and so on. If we carefully plan this strategy, one can envision the efficiency of allocating the budget towards recruiting initial adopters.

To optimize this strategy, we must consider the cost of convincing these initial adopters. More concretely, let $\mathcal{V}$ be the set of individuals, and suppose there exists a function $C(\cdot)$ that maps set $S \subseteq \mathcal{V}$ to the cost of recruiting them. Also, let $f(\cdot)$ be the *influence function* that maps $S$ to the number of individuals that will be influenced by the end of the campaign. Letting $B$ denote the budget, the optimal strategy should be to pick the set of individuals $S^* = \arg\max_{S:C(S)\leq B} f(S)$.

This problem is referred to as *Influence Maximization*, introduced in the celebrated paper of Kempe et al. [3] along with hardness results and approximation algorithms. Furthermore, their work further serves as a well-known exposition of the the so-called Independent Cascade (**IC**) and Linear Threshold (**LT**) *influence models*. Notably, they showed the resulting influence function induced in these models is a sub-modular function[1]. Of course, in real-world applications we usually do not know beforehand the structure of the *influence network*[2] or the influence process governing it, which naturally demands the non-trivial task of *learning* the influence function. Naturally, much interest was generated towards modeling the kind of data (and finding resulting sample complexities, thereof) required to recover the parameters that parameterize the influence function. In this survey we primarily concern ourselves with the following question:

*How and when are the underlying parameters of such a function learnable?*

In other words, through which algorithms and under what assumptions can we accurately estimate the influence function? This topic is recently formalized in [13] using the traditional PAC-learnability framework of Valiant [1], but is also closely coupled to progress in learnability-adjacent problems. In particular, the problem of recovering network and influence process structure via sampling was initiated by [6, 10], where [10] is notable for the first sample complexity upper and lower bounds for the widely-used IC model. Moreover, [6, 10] popularized a technique to decouple

[1]—which in themselves admit a host of independent interest and optimization methods [9]

[2]For example, we may know the presence of social links, but do not know anything about the relative "strength" of a social link.

the IC process parameter estimation problem into a node-wise (parallel) convex formulation. In [9], the authors established so-called Probably-Approximately-Mostly-Correct (PMAC) learnability for classes of submodular functions, of which contain influence functions induced by the IC and LT models [3]. Motivated to design a practical model-free algorithm for directly estimating the influence function (and circumventing the initial problem of learning the underlying influence process), the well-known InfluLearner algorithm is established in [12]. Recently, [15] revisited the problem of IC parameter process estimation [6, 10], and offered a hyperparametric approach provably lessening sample complexity and enjoying empirical success on real datasets.

**1.1 Organization** Through this survey, we hope to communicate an exposition and comparison on the common modeling assumptions in this space, with particular focus on PAC-style guarantees and algorithms collectively considered by [10, 12, 13, 15]. In Sec. 2 we give some definitions and common notation to apply to the surveyed works. In Sec. 3, 4, 5, and 6, we report the modeling assumptions, algorithms, and main contributions of each of [10, 12, 13, 15]. In Sec. 7, we attempt to reproduce experiments of [10, 12, 15].

## Contents

## 2 Preliminaries

**2.1 Notation** We denote a graph $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$. Let $n := |\mathcal{V}|$. Assume the edge set is directed and without self-loops. Let $\Gamma_{in}(u)$ denote the in-neighborhood of a node $u \in \mathcal{V}$. Denote $[k] = \{1, \ldots, k\}$ for $k \in \mathbb{N}$. We use the Landau notation in the standard way, and reserve $\tilde{\mathcal{O}}(\cdot)$ to hide logarithmic factors.

**2.2 Social Network Graphs** Consider a social network represented as a graph $G$, where nodes correspond to individuals, and edges correspond to social links. Whenever clear, 'node' or 'individual' or 'vertex' interchangeably describe an element of $\mathcal{V}$. By endowing $G$ with edge weights, we arrive at our first definition.

**Definition 1** (Influence Network)**.** *An influence network is a social network $G = (\mathcal{V}, \mathcal{E})$, where each edge $(u, v) \in \mathcal{E}$ is associated with a scalar "weight" $p_{uv} \in \mathbb{R}^+$.*

**Remark 1.** *The weight $p_{uv}$ should be interpreted as the weight of $u$'s influence upon $v$. Without loss of generality, we can assume these weights are restricted to the $[0, 1]$ interval.*[3]

The above captures the aspect of *how strongly* individuals affect each other. Now, we consider a model of how individuals conform to others' opinions—a diffusive *influence process* which governs the temporal shifting of "opinions" throughout an influence network.

**2.3 Influence Models and Influence Processes** We discuss two of the most popular influence models: the Discrete-Time Linear Threshold (LT) model, and the Discrete-Time Independent Cascade (IC) model, popularly exposited in [3]. The interested reader will also find continuous-time extensions of the IC model [8, 11].

Both models subsume processes defined for time $\tau = 0, \ldots, n-1$ on a given influence network $G = (\mathcal{V}, \mathcal{E})$. Furthermore, assume each node $u \in \mathcal{V}$ holds an *opinion* in $\{0, 1\}$ (e.g. this attribute may model whether an individual has adopted a product, or their preference for a political candidate, etc). The *seed set* is the subset $S \subseteq \mathcal{V}$ of nodes that start with opinion '1' at $\tau = 0$. If the opinion of node $u$ changes at time $\tau$, then it is said that $u$ is *influenced* at time $\tau$. Let $A_{\tau-1}$ denote the nodes influenced exactly at time $\tau - 1$, with $A_0 := S$. We call the set $A_{\tau-1}$ the *active* set at time $\tau$; only these nodes are allowed influence nodes at $\tau$. It is assumed that once a node changes opinion from '0' to '1', they may never revert. We distinguish the models accordingly:

1. **Linear Threshold (LT)** - The motivation in this model is that an individual's opinion can be determined by a linear combination of its familiars' opinions (e.g. an individual decides to buy tickets to a concert once enough of their friends are going).

   Concretely, each node $v \in \mathcal{V}$ holds a threshold $r_v \in \mathbb{R}^+$ such that $v$ adopts opinion '1' if $\sum_{u \in \Gamma_{in}(v) \cap A_{\tau-1}} p_{uv} \geq r_v$. Note that we can assume the thresholds are $[0, 1]$ (as the weights are), for we can always normalize the collection of weights and thresholds by the max.

2. **Independent Cascade (IC)** - Here, the motivation is that influence could be considered stochastically (e.g. the same individual as above may not deterministically go to the concert if all their friends are going, but they are more likely).

   The process is as follows. Interpreting the edge weights of $\mathcal{E}$ as probabilities, at each time step $\tau$ a node $v$ adopts the opinion of nodes $u \in \Gamma_{in}(v) \cap A_{\tau-1}$ independently with probability $p_{uv}$.

[3] Indeed, we can always normalize the weights by the max.

We call the *influence model* $\mathcal{M}$ as the collection of all influence processes $\mathcal{P}$ (i.e. all configurations of parameters) under that model. Due to the popularity of the IC over the LT model, we will accordingly compare results relating to the former. **For the rest of this survey, influence process always refers to an IC process, unless otherwise specified**.

**2.4 Influence Functions** If we fix an influence process $\mathcal{P}$ upon a influence network $G$, we have induced a so-called influence function which outputs a measurement of influence upon input of a seed set. We distinguish two definitions[4] encountered across the surveyed works.

**Definition 2** ("Expected Count" (**EC**) Influence Function, [3])**.** *Given an influence network and a influence process, an **EC** influence function $f : 2^{\mathcal{V}} \to \mathbb{R}_+$ maps a seed set $S$ to the (expected) number of influenced nodes.*

**Remark 2.** *The expectation is w.r.t. the (potential) randomness in the influence process, as well as the initial distribution of the seed.*

**Definition 3** ("Tabular Probabilities" (**TP**) Influence Function [13])**.** *Given an influence network and a influence process, a **TP** influence function $f : 2^{\mathcal{V}} \to [0,1]^n$ maps an seed set $S$ to a "influence probabilities" vector $f(S) := (f^{(1)}(S), \ldots, f^{(n)}(S)) \in [0,1]^n$, where $f^{(u)}(S)$ indicates the probability $u \in \mathcal{V}$ is influenced at any time $\tau$.*

**Remark 3.** *If the process is deterministic, the components are simply binary indicators.*

**2.5 Observation Model** In the real world, when we desire the influence function for an application (e.g. influence maximization), often the training data for our model consists of traces of the influence process, i.e. past information on *which* nodes were influenced. Here the observational model is un-unified. Some consider the $m$ training samples void of temporal information—e.g. lack of influence timestamps—while other works are more optimistic. Indeed, in some applications samples have rich temporal information, e.g. tracking a quick onset disease using the clinic check-in dates of patients. However, time data may not be reliable, e.g. the aforementioned disease has highly varied onset time. For each work, we clarify which observational model is used, although we define a commonly used model here.

**Definition 4** (Partial observation)**.** *The samples* **S** *are said to be under partial observation if* **S** *is a sequence of $m$ tuples $\{(S_i, \mathcal{I}_i)\}_{i \in [m]}$, each drawn i.i.d. from a distribution $\mathcal{D}$, where each tuple is referred to as a "cascade". Here $S_i \subseteq \mathcal{V}$ is a seed set drawn i.i.d from an initial distribution $\mathcal{D}_0$, and $\mathcal{I}_i \subseteq \mathcal{V}$ is the resulting influenced set (containing $S_i$ as well) sampled from the distribution induced by $\mathcal{D}_0$ and the influence process throughout $\tau = 0, \ldots, n-1$.*

# 3 Learning the Graph of Epidemic Cascades

In Netrapalli et al. [10], the authors studied the problem of recovering the parameters of an influence network, given samples resulting from the corresponding IC process. This problem was considered prior by [6], but their technique resulted in an NP-hard combinatorial optimization without sample complexity guarantees.

[4]—with our own terminology.

**Goal** An influence network $G = (\mathcal{V}, \mathcal{E})$ has unknown weights and edges. For the node $u \in \mathcal{V}$, the problem is to estimate $\Gamma_{in}(u)$ (parents of node $u$). The authors consider the estimate $\hat{\Gamma}_{in}(u)$ as error-free if it has:

(a) no false neighbors, i.e. $\hat{\Gamma}_{in}(u) \subseteq \Gamma_{in}(u)$

(b) all the "strong" neighbors, meaning all $j : p_{ji} > \frac{8}{\alpha}(2^{2\eta} - 1)$ are in $\hat{\Gamma}_{in}(u)$. Here $\eta$ is an algorithm parameter (See Additional Assumptions below for $\alpha$).

**Observational Model** Each of the $i = 1, \ldots, m$ samples is a "influence timestamps" vector $T_i \in (\mathbb{R}_+ \cup \{0, \infty\})^n$, where element $(T_i)_u$ is the time at which node $u$ was influenced. If a node $u$ is never influenced in the $i^{th}$ sample, $(T_i)_u$ is defined as $\infty$. Each sample is drawn i.i.d. from a distribution $\mathcal{D}$—characterized by $G$'s corresponding IC process—as well as the seed distribution $\mathcal{D}_0$ resulting from picking each node with fixed probability $p_{init}$.

**Algorithm** A sample $T_i$ is a vector whose components are random variables with a joint distribution determined by the edge weights $p_{uv}$, thus the authors advocate a traditional ML estimator to identify the graph most likely to have generated the samples, further leveraging a decoupling technique to express it as $n$ convex optimization problems (solvable in parallel).

The construction is as follows. Define $\theta_{uv} = -\log(1-p_{uv})$ for all $(u, v) \in \mathcal{E}$ (noting $\theta_{uv}$ increases monotonically with $p_{uv}$), and $\theta^*_u := \{\theta_{vu} : v \in \mathcal{V}\}$ as the set of parameters corresponding to the possible parents of $u$ (potentially any other node). Let $\theta$ be the collection of all parameters. The *log-likelihood* of a sample $T$ is defined as $\mathcal{L}(t; \theta) = \log \mathbb{P}_\theta[T = t]$. With an appropriate change of variables, the authors express the (convex) function as

$$\mathcal{L}(t; \theta) = \log(p^s_{init}(1 - p^s_{init})^{n-s}) + \sum_u \mathcal{L}_u(t; \theta^*_u) \tag{1}$$

where $s$ is the size of the seed set and we have the individual node terms

$$\mathcal{L}_u(t; \theta^*_u) := -\sum_{v:t_v \leq t_u - 2} \theta_{vu} + \log\left(1 - \exp\left(-\sum_{v:t_v = t_u - 1} \theta_{vu}\right)\right) \tag{2}$$

The algorithm 1 simply optimizes the summation of Eq. (2) (across all samples) to output a predicted neighborhood for each node (Note that solving Step 1 for all nodes allows one to recover $\hat{p}_{uv}$, which can be used to simulate influence of any desired set).

The first term of Eq. (1) accounts for the likelihood that a seed of size $s$ was realized, and the latter term aggregates the individual likelihoods. For a fixed node $u$ in Eq. (2), the latter term $-\sum_{v:t_v \leq t_u - 2} \theta_{vu}$ discounts the likelihood function for the parameters corresponding to nodes two or more time steps influenced prior to node $u$ (recall for a time $\tau$, the active nodes are those influenced at $A_{\tau-1}$). Of course, the other term of Eq. (2) is accounting for the parameters corresponding to influence from nodes immediately one time step before $u$ is influenced. Interestingly, the authors construct this function by considering a Markovian vector process $X(\tau)_{\tau=0,\ldots,n-1}$ where

$$X_u(\tau) = \begin{cases} 0, & \text{if } u \text{ has opinion '0' at time } \tau \\ 1, & \text{if } u \text{ has opinion '1' and is active at time } \tau \\ 2, & \text{if } u \text{ has opinion '1' and is inactive at time } \tau \end{cases}$$

This process has the property that each of its sample paths are in bijection with a particular influence timestamps vector—hence, they have the same probability measure. Expressing the

likelihood function in terms of the vector process, the derivation leverages the Markovian property to factorize the likelihood function via chain rule.

**Algorithm 1** ML Algorithm for node $u$, Netrapalli et al.

**Require:** $\eta > 0$, samples $\mathbf{S} = \{T_i\}_{i\in[m]}$

1: Solve $\hat{\theta} := \arg\max_{\theta_u^*} \sum_{i=1}^{m} \mathcal{L}_u(T_i; \theta_u^*)$

2: Choose in-neighborhood by threshold: $\hat{\Gamma}_{in}(u) := \{v : \hat{\theta}_{vu} \geq \eta\}$. Output $\hat{\Gamma}_{in}(u)$.

**Additonal Assumptions** The influence network satisfies the existence of $\alpha > 0 : \sum_v p_{vu} < 1 - \alpha$ for every node $u$, which is used to enforce the property called *correlation decay*—implying the probability that node $u$ is influenced at time $\tau$ decays as $\mathcal{O}((1-\alpha)^\tau)$, hence the process does not travel too far from the seed. This property is essential for their sample complexity guarantees, but their experiments without it seem to suggest the same result. They conjecture the same result holds without correlation decay—to our knowledge not yet proven.

**Sample Complexity** Let $d_u$ denote the in-degree of $u \in \mathcal{V}$. For any $\delta \in (0,1)$, Algorithm 1 admits sample complexity $\mathcal{O}(d_u \log(\frac{n}{\delta}))$ for outputting an error-free $\hat{\mathcal{V}}_u$ w.p. $\geq 1-\delta$. Letting $d := \max_u d_u$ and $\delta := \epsilon/n$, a union bound translates this to error-free recovery of all neighborhoods w.p. $\geq 1-\epsilon$ in $\mathcal{O}(d^2 \log(\frac{n^2}{\epsilon}))$ samples.

The authors provide an information-theoretic lower bound as follows. Letting $\mathcal{G}$ denote the collection of all graphs for a particular set of edge weights, let $G \in \mathcal{G}$ be chosen uniformly at random. Suppose our algorithm outputs $\hat{G}$ upon observing samples $(T_i)_{i\in[m]}$. Let $\mathcal{B}(G') \subseteq \mathcal{G}$ be a set of pre-defined graphs for every graph $G' \in \mathcal{G}$. We say the estimate is error-free if $G \in \mathcal{B}(\hat{G})$. For example, if we let $\mathcal{B}(G')$ denote the $r$ edit-distance ball centered around $G'$, then we say the output of the algorithm is error-free if the true graph $G$ is within the ball $\mathcal{B}(\hat{G})$. Letting $P_e := \Pr[\, G \notin \mathcal{B}(\hat{G}) \,]$, and $T$ as the underlying random variable which generates the samples, If

$$m \geq \frac{(1-P_e)\log \frac{|\mathcal{G}|}{\sup_{G'} |\mathcal{B}(\mathcal{G}')|} - 1}{\sum_{u\in\mathcal{V}} H(T_u)}$$

then we have a probability of error at most $P_e$.

## 4 Influence Function Learning in Information Diffusion Networks

According to Du et al. [12], the folklore approach to learn the **EC** influence function is to first estimate the parameters of the underlying influence network (e.g. through the method above), then obtain an estimate of the influence function for a desired set $S$ through repeatedly sampling using the estimated parameters. However, if one does not select the proper influence model, e.g. the actual process generating the samples is LT, then this method may break. Therefore, they argue that a proper approach should be influence model-free—motivating their approach of estimating the true **EC** influence function directly. For the remainder of this section, all influence functions are understood to be **EC** influence functions.

**Goal** Given an influence network $G = (\mathcal{V}, \mathcal{E})$, and any influence process in any influence model[5], the goal is to estimate the resulting **EC** influence function, for any subset $S \subseteq \mathcal{V}$, with as few

[5]The authors assume less structure here. For instance, the influence model isn't restricted to what we have described. For comparison-sake we focus on this setting.

samples as possible.

**Observational Model** Samples $\mathbf{S}$ is under partial observation (Def. 4).

**Algorithm** Let $\phi(x) := \min\{x, 1\}$ for all $x \in \mathbb{R}$. Denote $\chi_S$ as the $n$-dimensional binary incidence vector for the seed set $S$, i.e. $(\chi_S)_i = 1 \iff i \in S$. The authors take the view that the influence network and process induce a distribution over the set of reachability matrices $\mathbf{R} \in \{0,1\}^{n \times n}$ where $\mathbf{R}_{ij} = \mathbb{1}[\text{node } i \text{ is reachable from node } j]$. Therefore, having sampled a $\mathbf{R}$ according to some distribution $p_{\mathbf{R}}$, one can compute an indicator of whether the seed $S$ eventually caused the influence of a node $j$ as $\phi(\chi_S^\top \mathbf{R}_{:,j})$ (define $\mathbf{R}_{:,j}$ as column $j$ of $\mathbf{R}$). It follows that the influence function evaluated at a seed $S$ is given by

$$\sigma(S) = \mathbb{E}_{\mathbf{R}\sim p_{\mathbf{R}}}[\sum_{j\in\mathcal{V}} \phi(\chi_S^\top \mathbf{R}_{:,j})] = \sum_{j\in\mathcal{V}} \mathbb{E}_{\mathbf{R}\sim p_{\mathbf{R}}}[\phi(\chi_S^\top \mathbf{R}_{:,j})] := \sum_{j\in\mathcal{V}} \Pr[\phi(\chi_S^\top \mathbf{R}_{:,j}) = 1 \mid \chi_S] \tag{3}$$

Consider the summand term,

$$f_j(\chi_S) := \Pr[\phi(\chi_S^\top \mathbf{R}_{:,j}) = 1 \mid \chi_S] = \mathbb{E}_{r(j)\sim p_r(j)}[\chi_S^\top r(j)] \tag{4}$$

where $r(j) := \mathbf{R}_{:,j}$, and $p_r(j)$ is the marginal distribution of the $j^{th}$ column of $\mathbf{R} \sim p_{\mathbf{R}}$. How can we learn these functions? Let $y_{ij} := \mathbb{1}[\text{node } j \text{ is in set } \mathcal{I}_i]$. One can observe that the conditional likelihood function of node $j$ being influenced according to the sample $(S_i, \mathcal{I}_i)$ (conditioned on the seed $S_i$) is,

$$f_j(\chi_{S_i})^{y_{ij}}(1 - f_j(\chi_{S_i}))^{1-y_{ij}} \tag{5}$$

But rather than directly maximize this objective, the author's use the following technique: Let $q_j$ be a vector, with entry $(q_j)_i$ as the fraction of occurrences across all $m$ samples (empirical frequency) that $j$ was in the influence set when $i$ was in the seed set. $q_j$ qualifies as a distribution on $\{0,1\}^n$, and further serves as a surrogate for $p_r(j)$. Having drawn $K$ random vectors $r_1, \ldots, r_K$ from $q_j$, one can express $f_j(\chi_S)$ as a convex combination of these vectors,[6]

$$f^w(\chi_S) = \sum_{k=1}^{K} w_k \phi(\chi_S^\top r_k) \quad \text{s.t.} \quad \sum_{k}^{K} w_k = 1, w_k \geq 0 \quad \forall k \in [K] \tag{6}$$

If $K$ is sufficiently large—then there exists a $w = (w_1, \ldots, w_k)$ such that $f^w(\chi_S)$ is a good approximation of $f_j(\chi_S)$ (See Lemma 1, [12]). Therefore, one can maximize the likelihood objective function with $f^w(\chi_{S_i})$ in place of $f_j(\chi_{S_i})$ in Eq. (5). However, one may implicitly condition on a zero probability event without any further modifications. A final modification substitutes $f^w(\chi_S)$ for the Winsorized surrogate $f^{w,\lambda}(\chi_S) := (1-2\lambda)f^w(\chi_S) + \lambda$ for some $\lambda \in (0, 1/4)$ (truncated to range $[\lambda, 1-\lambda]$).

Finally, this yields the log-likelihood function for training set $\mathcal{S}$,

$$\mathcal{L}_j(\mathcal{S}; w) = \sum_{i=1}^{m} \Big( y_{ij} \log f^{w,\lambda}(\chi_{S_i}) + (1 - y_{ij}) \log(1 - f^{w,\lambda}(\chi_{S_i})) \Big) \tag{7}$$

To maximize this, the authors advocate the popular exponentiated gradient (EG) algorithm in [2].

**Additional Assumptions** The distribution $p_j$ is fully factorized, i.e. a product measure w.r.t. its component distributions.

[6] This technique is similar to the idea of random kitchen sinks [5], and is used to expedite training.

**Algorithm 2** InfluLearner

**Require:** Training set $\mathbf{S} = (\mathcal{S}_i, \mathcal{I}_i)_{i=1,\dots,m}$, $\lambda \in \left(0, \frac{1}{4}\right)$

1: **for** each node $j \in [d]$ **do**
2: Sample $K$ random features $r_1, \dots, r_K$ from $q_j$
3: Compute $\Phi(\mathcal{S}i) := (\phi(\chi_{S_i}^\top r_1), \dots, \phi(\chi_{S_i}^\top r_K))$ for all $i \in [m]$.
4: Initialize $w_1 = (\frac{1}{K}, \frac{1}{K}, \dots, \frac{1}{K})$
5: Solve $\hat{w} := \arg\max_w \mathcal{L}_j(\mathcal{S}; w)$ using EG Algorithm
6: $\hat{f}_j^{w,\lambda}(\chi_S) = \lambda + (1 - 2\lambda)\hat{w}^\top \Phi(\chi_S)$
7: **end for**
8: Output $\hat{\sigma}(S) = \sum_{j=1}^{d} \hat{f}_j^{w,\lambda}(\chi_S)$

**Sample Complexity** Setting $\lambda = \tilde{\mathcal{O}}(\frac{\epsilon}{d})$, $K = \tilde{\mathcal{O}}(\frac{d^2}{\epsilon^2})$, and $m = \tilde{\mathcal{O}}(\frac{d^3}{\epsilon^3})$, then

$$\mathbb{E}_{\mathcal{S}\sim\mathcal{D}^m, S\sim\mathcal{D}_0}[\,(\hat{\sigma}(S) - \sigma(S))^2\,] \leq \epsilon \quad \text{w.p. } \geq 1 - \delta \tag{8}$$

# 5 Learnability of Influence in Networks

Although Narasimhan et al. [13] was not the first work to consider the formal learnability of the class of influence functions (See Balcan et al. [9]), they were able to establish traditional PAC learnability specifically for different classes of influence functions induced by various influence models; in particular—the IC model.

**Goal** Establish the PAC-learnability for the class of **TP** influence functions induced by the IC model (amongst others). Under a social network graph $G = (\mathcal{V}, \mathcal{E})$, where only the edge set (but not the weights) is known. Let $\mathcal{F}_{G,\mathcal{M}}$ denote the class of all **TP** influence functions imposed by each influence process $\mathcal{P} \in \mathcal{M}$ upon $G$, e.g. $\mathcal{F}_{G,\text{IC}}$ is the class of such functions under the IC model.

**Definition 5** (Loss function, Narasimhan et al.)**.** *For a set of influenced nodes $Y \subseteq \mathcal{V}$, and any probability vector $p \in [0,1]^n$, the loss function $\ell : 2^{\mathcal{V}} \times [0,1]^n \to \mathbb{R}_+$ assigns a real-value to $\ell(Y, p)$ to quantify the discrepancy between $Y$ and $p$.*

For $\mathcal{F}_{G,\text{IC}}$, the authors appropriately consider the squared loss,

$$\ell_{sq}(Y, p) := \frac{1}{n} \sum_{u \in \mathcal{V}} \mathbb{1}[u \in Y](1 - p_u)^2 + \mathbb{1}[u \in Y]p_u^2$$

**Definition 6** (Error function, Narasimhan et al.)**.** *For $f \in \mathcal{F}_{G,\mathcal{M}}$ and seed distribution $\mathcal{D}_0$, the error of $f$ w.r.t. loss $\ell$ is $err^\ell[f] = \mathbb{E}[\,\ell(Y, f(X))\,]$, where the expectation is over a random drawing of seed $X$ from $\mathcal{D}_0$, and the randomness in the influenced set $Y$.*

**Definition 7** (PAC-learnability, Narasimhan et al.)**.** *The class $\mathcal{F}_{G,\mathcal{M}}$ is said to be PAC-learnable w.r.t. loss $\ell$ if there exists an algorithm $\mathcal{A}$ such that for all $\mathcal{P} \in \mathcal{M}$ (i.e. all parametrizations of the model), and for a subset of all distributions $\mathcal{D}_0$ over the seed sets: whenever the input to $\mathcal{A}$ is a training set $\mathbf{S}$ under some observational model, with $m \geq poly(\frac{1}{\epsilon}, \frac{1}{\delta})$, $\mathcal{A}$ returns a function $f \in \mathcal{F}_{G,\mathcal{M}}$ satisfying*

$$\Pr[\, err^\ell[f] - \inf_{f \in \mathcal{F}_{G,\mathcal{M}}} err^\ell[f] \geq \epsilon\,] \leq \delta$$

*where the probability measure is w.r.t. the randomness in the sample* $\mathbf{S}$*. If the algorithm admits this guarantee with sample complexity polynomial in* $m$ *and* $|\mathcal{V}| = n$*, then* $\mathcal{F}_G$ *is said to be efficiently PAC-learnable.*

**Remark 4.** *We emphasize* $\mathcal{F}_{G,\mathcal{M}}$ *is said to be PAC-learnable w.r.t. the observational model.*

**Observational Model** The authors consider both the partial observations model (Def. (4)), as well as the following:

**Definition 8** (Full observations, Narasimhan et al.)**.** *The samples* $\mathbf{S}$ *is said to be under full observations if* $\mathbf{S}$ *is a sequence of* $m$ *tuples* $\{(S_i, \mathcal{I}_{i,(0:n-1)})\}_{i\in[m]}$*, drawn i.i.d. from a distribution* $\mathcal{D}$*. Here* $S_i \subseteq \mathcal{V}$ *is a seed set drawn i.i.d from an initial distribution* $\mathcal{D}_0$*, and* $\mathcal{I}_{i,(0:n-1)} = \{\mathcal{I}_{i,(0)}, \mathcal{I}_{i,(1)}, \ldots, \mathcal{I}_{i,(n-1)}\}$ *is the sequence of sets influenced exactly at timestep* $\tau = 0, \ldots, n-1$ *for the* $i^{th}$ *cascade. Note* $\bigcup_{\tau=0}^{n-1} \mathcal{I}_{i,(\tau)}$ *is the set of all influenced nodes for the* $i^{th}$ *sample—sampled from the distribution induced by* $\mathcal{D}_0$ *and the influence process throughout.*

**Algorithm** First, we describe the PAC-learning algorithm that the authors use to prove the learnability of $\mathcal{F}_{G,\text{IC}}$. As an overview, the proof pipeline is as follows: (i) they derive a closed-form expression $f \in \mathcal{F}_{G,\text{IC}}$, and show Lipschitzness of $f$; (ii) specify the learning algorithm; (iii) construct $\epsilon$-cover over space of processes $\mathcal{P}$ (i.e. all edge weight configurations); (iv) translate cover on $\mathcal{P}$ to a cover on $\mathcal{F}_{G,\text{IC}}$ via Lipschitzness; (v) apply known uniform convergence argument for covering numbers between estimated parameters and true parameters to bound error between estimate and true $f$.

Consider (i). For any set $X$, and $\mathbf{p}$ as the vector cataloging all the $p_{uv}$'s (for a particular influence process $\mathcal{P}$) one can express the **TP** influence function $f^{\mathbf{p}}(X)$ through its components,

$$f_u^{\mathbf{p}}(X) = \sum_{A\subseteq\mathcal{E}} \prod_{(u',v')\in A} p_{v'u'} \prod_{(u',v')\notin A} (1 - p_{v'u'}) \cdot \sigma_u(A, X)$$

where

$$\sigma_u(A, X) := \mathbb{1}[u \text{ is reachable from } X \text{ via edges in } A]$$

Observing that the summation is over the powerset of $\mathcal{E}$, one can see this computation is simply summing all the possible paths $X$ may influence $u$, and weighting each by its corresponding probability. The authors prove (See Lemma 1, [13]) that for all sets $X$, and all $u \in \mathcal{V}$,

$$||\mathbf{p} - \mathbf{p}'||_1 \leq \epsilon \implies |f_u^{\mathbf{p}}(X) - f_u^{\mathbf{p}'}(X)| \leq \epsilon \quad \forall \mathbf{p}, \mathbf{p}' \in [0,1]^{|\mathcal{E}|}$$

Consider (ii). Again, one can take an ML-based approach, defining for each partial observations sample $s = (X, Y)$,

$$\mathcal{L}(X, Y; \mathbf{p}) = \sum_{u\in\mathcal{V}} \mathbb{1}[u \in Y] \log(f_u^{\mathbf{p}}(X)) + (1 - \mathbb{1}[u \in Y]) \log\left(1 - f_u^{\mathbf{p}}(X)\right) \tag{9}$$

and simply choose $\hat{\mathbf{p}}$ that maximizes the sum of all such likelihoods $\sum_{i=1}^{m} \mathcal{L}(S_i, \mathcal{I}_i; \hat{\mathbf{p}})$. For (iii), one can construct a cover of size $\mathcal{O}\big((\frac{|\mathcal{E}|}{\epsilon})^{|\mathcal{E}|}\big)$ over the space of possible IC parameters, which gives a cover of the same size in the function space. (iv-v) yields PAC-learnability for the partial observations setting.

Under full observations, one can follow a decoupled (localized) algorithm in spirit of [10] (since observations are have time information) to estimate the edge probabilities, then: (i) construct an $\epsilon$-cover in the space of local log-likelihood functions, showing the evaluation at the estimated edge

probabilities are not too far from optimal; (ii) argue the expected log-likelihood is strongly concave under Assumption 1, which translates closeness in local log-likelihood to closeness in the parameter space; (iii) and finally translate (via the aforementioned Lipschitzness) this closeness in parameter space to closeness in the influence function space $\mathcal{F}_{G,\text{IC}}$.

**Additional Assumptions** Assume that all $p_{uv}$'s are bounded away from 0 and 1, i.e. $\exists \lambda > 0 : p_{uv} \in [\lambda, 1-\lambda] \quad$ for all $(u,v) \in \mathcal{E}$.

**Assumption 1** (Narasimhan et al.)**.** *In addition to the above, assume* $\prod_{v \in \Gamma_{in}(u)}(1-p_{vu}) \geq \lambda$ *for all* $u \in \mathcal{V}$*. Also, each node in* $\mathcal{V}$ *is chosen independently in the initial seed set with probability* $p_{init} \in (0,1)$*.*

**Sample Complexity**

1. The class of IC influence functions $\mathcal{F}_{G,\text{IC}}$ is PAC-learnable under partial observations, w.r.t. the square loss $\ell_{sq}$, admitting sample complexity $\tilde{\mathcal{O}}(\frac{n^3|\mathcal{E}|}{\epsilon^2})$. Under full observations and Assumption 1, $\mathcal{F}_{G,\text{IC}}$ is efficiently PAC-learnable (w.r.t same loss) and admits sample complexity $\tilde{\mathcal{O}}(\frac{n|\mathcal{E}|^3}{\epsilon^2})$.
2. For the ML objective in the partial observation setting, for any $(\epsilon, \delta) \in (0,1)^2$, with at least $\tilde{\mathcal{O}}(\frac{n^3|\mathcal{E}|}{\epsilon^2})$ samples,

$$\sup_{\mathbf{p} \in [\lambda, 1-\lambda]^{|\mathcal{E}|}} \mathbb{E}[\, \frac{1}{n}(\mathcal{L}(X,Y;\mathbf{p}) - \mathcal{L}(X,Y;\hat{\mathbf{p}}))\,] \leq \epsilon \quad \text{w.p.} \geq 1-\delta \tag{10}$$

The authors point out that the discrepency between sample complexities in the partial vs. full observations setting (c.f (1)), i.e. linear dependence on $n$ rather than cubic, is the savings in performing local ML estimation. However, the cubic dependence on $|\mathcal{E}|$ is incurred by their proof technique, which requires estimating the parameters first, then translating this to a guarantee on the resulting influence functions.

## 6 Learning Diffusion using Hyperparameters

Prior to the work of Kalimeris et al. [15], all previous formulations had taken the edge weights $p_{uv}$ as arbitrary—a key contributor of the high sample complexity for the local ML estimator based algorithms. Without correlated edge probabilities, intuitively one needs many more samples. [15] introduces a technique to structure these edge probabilities in the IC model.

As a demonstration (per [15]), consider a bipartite graph $G = (U, V, \mathcal{E})$, where $|U| = |V| = n$ and $\mathcal{E}$ is a perfect matching, so $|\mathcal{E}| = n$. Suppose we wish to estimate $\hat{p}_{uv}$ such that $|\hat{p}_{uv} - p_{uv}| \leq \epsilon$ holds for all edges. If we receive samples $\mathbf{S}$ under partial observation, then every $(S_i, \mathcal{I}_i)$ yields a realization of $n$ Bernoulli random variables $X_{uv}$, each having mean $p_{uv}$. In this manner, for each edge $(u,v)$, let $X_{uv}^{(1)}, \ldots, X_{uv}^{(m)}$ be the $m$ i.i.d. Bernoulli r.v.'s realized. By Hoeffding inequality $\Pr[\, |\frac{1}{m}\sum_{i=1}^{m} X_{uv}^{(i)} - p_{uv}| \geq \epsilon \,] \leq e^{-2m\epsilon^2}$. Taking $e^{-2m\epsilon^2} \leq \delta/n$ and applying a union bound, this yields a sample complexity of $\mathcal{O}(\frac{|\mathcal{E}|}{\epsilon^2}\log(\frac{|\mathcal{E}|}{\delta}))$.

By comparison, suppose each $p_{uv} = 1/(1+e^{-\mathbf{x}_{uv}^\top \theta})$, where $\theta$ is a hyperparameter in $\mathbb{R}^z$, and $\mathbf{x}_{uv}$ is a feature vector containing information about $u$ and $v$. Effectively, this parametrization is positing *homophily* upon the network, i.e. if $u$ and $v$ share certain characteristics, then their influence probability should be close to another pair of nodes $u'$ and $v'$ who share the same characteristics.

Now, learning $p_{uv}$'s is a logistic regression problem, which is known to admit sample complexity $\mathcal{O}(\frac{z}{\epsilon}\log(\frac{z}{\delta}))$. Hence, if for a general graph $|\mathcal{E}|$ is large and $z$ is small, then the sample complexity savings are significant.

**Goal** Consider some influence network $G = (\mathcal{V}, \mathcal{E})$—with known edges but unknown weights—under an IC process, where every edge $(u, v)$ is further endowed with a feature vector $\mathbf{x}_{uv} \in \mathbb{R}^d$. Suppose there exists a $\theta$ belonging to hypothesis class $\mathcal{H} = [-B, B]^z$ for some $B > 0$ such that $p_{uv} = p_{uv}(\theta) := 1/(1 + e^{-\mathbf{x}_{uv}^\top \theta})$. The goal is to bound the sample complexity of learning $\mathcal{H}$, i.e. guarantee that

$$\sup_{\theta \in \mathcal{H}} \mathbb{E}_{\mathbf{S}\sim\mathcal{D}^m}[\,\mathcal{L}(\mathbf{S}, \theta)\,] - \mathbb{E}_{\mathbf{S}\sim\mathcal{D}}[\,\mathcal{L}(\mathbf{S}, \hat{\theta})\,] \leq \epsilon \quad \text{w.p. } \geq 1 - \delta \tag{11}$$

where $\mathbf{S}$ is the set of samples drawn i.i.d. from some distribution $\mathcal{D}$ (described below).

**Observational Model** The manner in which $\mathbf{S}$ is drawn differs from other works. For a influence network $G$ under the corresponding IC process, consider a realization of a cascade $B_0, B_1, \ldots, B_{n-1}$, where $B_\tau$ is set of nodes influenced exactly at time $\tau$, and $B_0$ is sampled via some initial distribution $\mathcal{D}_0$. Having defined $C_1, \ldots, C_{\tau-1}$, define the collection of labels

$$C_\tau = \{((B_{\tau-1}, v), y) \mid v \text{ within 1-hop of } B_{\tau-1},\ v \notin \bigcup_{i=1}^{\tau-1} B_i,\ y = \mathbb{1}[v \in B_\tau]\}$$

which categorizes all the not-yet influenced nodes within 1-hop of $B_{\tau-1}$: '1' if they are influenced at the next time-step, and '0' otherwise. The authors assume that the sample given to the algorithm is drawn uniformly at random from the resulting union of all such collections for that cascade. Informally, each sample involves a realization of a cascade, but the actual information supplied to the algorithm is just one of the many possible indicators of whether some node was influenced in that cascade at some time instant. At first glance, one may percieve that the resulting model is more "instructive" than the partial observations model, since over 1-hop we do not have to worry about which set was active when a candidate node was influenced, as in the partial observation model. However, realize that for a single sample $((X, v), 1)$, if $v$ has multiple parents in $X$, then it is difficult to distinguish which node of $X$ actually was the cause of $v$ becoming influenced.

**Algorithm** The authors view each event $\{v \text{ is influenced by set } X \text{ given } \theta\}$ as a Bernoulli r.v. with success probability $f_v^\theta(X) := 1 - \prod_{u \in X \cap \Gamma_{in}(v)}(1 - p_{uv}(\theta))$ (probability the event "no parents of $v$ infects $v$" does **not** occur). Thus, the likelihood of sample $s = ((X, v), y)$ is $f_v^\theta(X)^y(1 - f_v^\theta(X))^{1-y}$. Hence, the respective log-likelihood of $s \in \mathbf{S}$ is

$$\mathcal{L}(s, \theta) = y \log(f_v^\theta(X)) + (1 - y)\log(1 - f_v^\theta(X))$$

So, to estimate the edge probabilities $p_{uv}$, we can solve $\hat{\theta} = \arg\max_{\theta \in \mathcal{H}} \frac{1}{m}\sum_{s \in \mathbf{S}} \mathcal{L}(s, \theta)$. However, as the authors note, this objective is not concave (recall the above example for sample $((X, v), 1)$—it is not clear how the estimate $\hat{\theta}$ should be updated).

To deal with this, let $\mathbf{S}_0$ be the set of all samples of the form $((X, v), 1)$ where $v$ has more than one parent in $X$. We can partition $\mathbf{S} = \mathbf{S}_0 \sqcup (\mathbf{S} \setminus \mathbf{S}_0)$. Our objective can be expressed as

$$\tilde{f}(\theta) := \frac{1}{m}\sum_{s \in \mathbf{S}_0} \mathcal{L}(s, \theta) + \frac{1}{m}\sum_{s \in (\mathbf{S} \setminus \mathbf{S}_0)} \mathcal{L}(s, \theta) := f(\theta) + \xi(\theta) \tag{12}$$

Here, $f(\theta)$ is concave. The authors suggest three approaches:

1. Optimize $f(\theta)$ directly via Gradient Descent, ignoring the challenging samples. Of course, then the output will have some extra error, but in certain cases that error can be bounded based on the probability $p_0$ of observing such samples.

2. Interpret the optimization of $\tilde{f}(\theta)$ as concave optimization under noise, and apply existing techniques thereof [14].

3. Use standard concave optimization techniques (e.g. SGD).

**Additional Assumptions** Assume that all $p_{uv}$'s are bounded away from 0 and 1, i.e. $\exists \lambda >: p_{uv} \in [\lambda, 1-\lambda] \quad$ for all $(u,v) \in \mathcal{E}$.

**Sample Complexity** We report the sample complexity for the suggested approach (1) for optimizing $\tilde{f}(\theta)$[7]. Let $m_0 := |\mathbf{S}|$. For any sample $s = ((X,v),y)$. Let $\Delta := max_{s\sim\mathcal{D}}|X \cap \Gamma_{in}(v)|$ denote a bound on the potential parents of any sample $s$, and let $\Delta_{\mathbf{S}}$ be the corresponding bound for realized training set $\mathbf{S}$. If $\frac{m_0}{m} \leq \frac{\epsilon}{\Delta_{\mathbf{S}} \log(1/\lambda)}$ (i.e. there are not too many challenging samples), then for any $(\epsilon, \delta) \in (0,1)$, whenever

$$m = \mathcal{O}\left(\frac{\Delta^2}{\epsilon^2}\left(z\log(\frac{zB}{\lambda^{\Delta}\epsilon}) + \log(\frac{1}{\delta})\right)\log^2(\frac{1}{\lambda})\right)$$

we have

$$\sup_{\theta\in\mathcal{H}} \mathbb{E}_{\mathbf{S}\sim\mathcal{D}^m}[\,\mathcal{L}(\mathbf{S},\theta)\,] - \mathbb{E}_{\mathbf{S}\sim\mathcal{D}}[\,\mathcal{L}(\mathbf{S},\hat{\theta})\,] \leq 3\epsilon \quad \text{w.p. } \geq 1-\delta$$

Observe that $m_0/m$ converges, for large $\mathbf{S}$ to the probability $p_0$ of seeing challenging examples. See Lemma 7, [15] for a bound on $p_0$.

# 7 Experiments

In this section we report the results of our attempts to reproduce the algorithms described in [10,12, 15]. To be able to compare papers consistently, we use Mean Absolute Error (MAE) as a common metric across all experiments. All our code for reproducing the experiments is available on GitHub at https://github.com/shngt/influence-pac.

## 7.1 Learning the Graph of Epidemic Cascades

Netrapalli et al.'s paper serves as a baseline for several others in this area, and thus is instructive to implement. However, their algorithm does not scale well to large networks, so we only test our implementation on a (relatively) small network as proof-of-concept. We created a synthetic Erdos-Renyi $G_{n,m}$ graph with 100 vertices and 250 edges. We ran the procedure for training sets of size 100 to 100,000, with 100 cascades for each seed set. In other words, we ran the procedure for 1 to 1,000 seed sets. The seed sets for both training and testing were generated with $p_{init} = 0.02$ i.e. the probability of including a particular node in the seed set. We use the training data to perform Maximum Likelihood estimation via Sec. 3 and estimate the $p_{uv}$'s. For evaluation, we generated the same number of random seed sets and simulated the influence process using the true weights and the predicted weights to obtain true and predicted influence. We ran the entire procedure 10 times for each training set size, and report the mean MAE in Fig. 1.

Interestingly, we found that the algorithm tended to produce large $\theta$'s, especially if there were not many samples for a particular edge, which meant several of the predicted $p_{uv}$'s were close to

[7] Approach (2) and (3) do not readily admit results on sample complexities.

1. This meant that the predicted influences were much higher than the actual influences, which resulted in much higher MAEs than in other experiments. Note that this is not an issue per the problem formulation, since it only aims to make sure false neighbours are excluded and strong neighbours are included. We believe that the MAE gap could be narrowed with the introduction of more data.

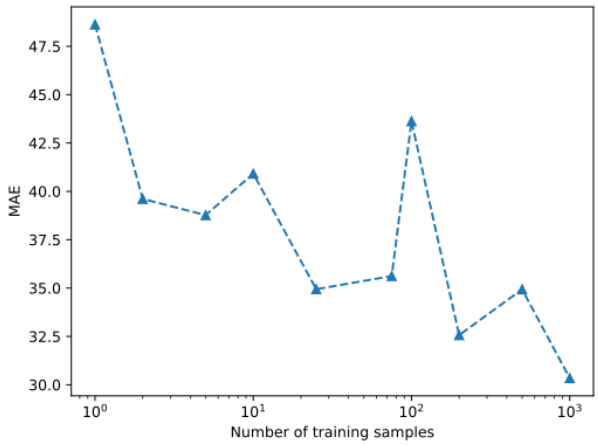


Figure 1: MAE vs. number of seed sets for reproduction of Netrapalli et al.

**7.2 Influence Function Learning in Information Diffusion Networks** We implemented Du et al.'s InfluLearner as described in the paper, but we found that the algorithm does not scale well without parallelization, a fact reported in the paper as well. This is mainly due to specific time-consuming steps in the training and evaluation stages such as estimating empirical probabilities during evaluation and having to perform several rounds of sampling during training. Steps like these are particularly inefficient in Python. The exponentiated gradient algorithm suggested by the authors is also highly unstable in this particular setting. For these reasons, we were unable to reproduce meaningful results on graphs of similar sizes as in our other experiments, but believe we should be able to reproduce their results after applying heavier parallelization and moving the more time-consuming subroutines to a more efficient language like C++ or Matlab.

**7.3 Learning Diffusion using Hyperparameters** The hyperparametrization approach in this paper greatly reduces the number of parameters to be learned and allowed us to experiment with larger graphs. The hyperparameter is taken as a vector $\theta$ of size 20 and for each node, we sampled a random feature vector of size 10 with values between 0 and 1. The true $\theta$ is randomly sampled with values between -1 and 1. We implemented the third approach suggested by the authors (i.e. solving $\tilde{f}(\theta)$ via SGD, see Sec. 6 and tested it on three datasets - a synthetic Erdos-Renyi graph with 1,000 vertices and 20,000 edges, a synthetic Kronecker graph with 1,024 vertices and 2,655 edges, and the real-world email-Eu-core dataset [4] with 1,005 vertices and 25,571 edges. Similar to the original paper, we ran the procedure for training sets of size 1 to 100,000, where a single sample consists of a tuple $((X, v), y)$. The seed sets for both training and testing were generated using a power law distribution with parameter 2.5. Note that while the original paper reports the average error in estimating $p_{uv}$, we focused on MAE instead as stated earlier. The graphs for MAE vs. training set size are in Fig. 2, Fig. 3 and Fig. 4 for each dataset. For evaluation, we generated 100,000 random seed sets and simulated the influence process using the true weights

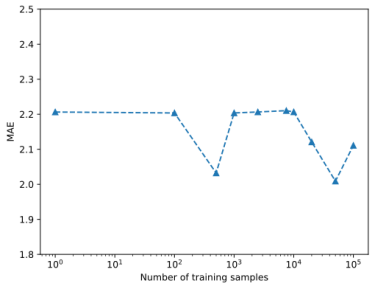


Figure 2: Erdos-Renyi

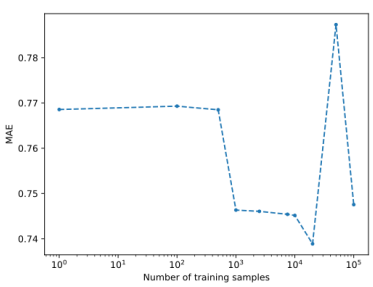


Figure 3: Kronecker

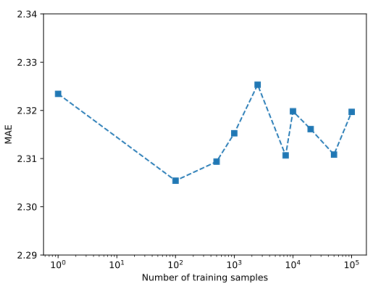


Figure 4: email-Eu-core

Figure 5: MAE vs. training set size for reproductions of Kalimeris et al.

and the predicted weights to obtain true and predicted influence. We ran the entire procedure 50 times for each training set size, and report the mean MAE in the corresponding graphs. For the SGD procedure, we take the learning rate as $\sqrt{T}$ where $T$ is the size of the training set, and all other parameters as 0. Also, since computing gradients of the likelihood is relatively involved, we leveraged PyTorch's auto-differentiation capabilities and an NVIDIA Titan X 12 GB GPU for acceleration.

# 8 Future Work

In this section we conclude with some suggestions for possible research directions.

(a) *A unified observational model*—The observational model has a large determination on the resulting sample complexity, with sharp contrast between the two dominating catagories of models, i.e. with and without time information. Moreover it is not unreasonable to suspect that a smooth transition between the two extremes can be characterized by an observational model that posits time-full information perturbed by structured additive noise, e.g. centered/bounded, sub-gaussian, exponential, etc. Possibly in such a model, tighter application-specific guarantees could be made.

(b) *Improvements on mentioned works*—For Netrapalli et al., we noted how the assumption of correlation decay may not be necessary. To our knowledge, the same guarantees have not been presented without this assumption, and may make for an interesting work (c.f. **Additional Assumptions** in Sec. 3). Other candidates for improvement include decreasing the cubic dependence on $|\mathcal{E}|$ from Narasimhan et al.'s sample complexity under full observation (c.f. **Sample Complexity** in Sec. 5)

(c) *Interventional Model*—In some applications we may be able to innoculate seeds into the studied social network, e.g. the product testing phase of a marketing campaign. An interesting generalization of the IC model could be to allow a budgeted intervention of the learner into the environment to lower sample complexity. Such a model could likely connect this field to the broader field of *causal discovery*.

## References

[1] Leslie G. Valiant. A theory of the learnable. Commununications of the ACM, 27(11):1134–1142, 1984.

[2] Jyrki Kivinen, and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. Information and Computation, 132(1):1–64, January 1997.

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

[4] Jure Leskovec, Jon M. Kleinberg. and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. ACM transactions on Knowledge Discovery from Data (TKDD), 1(1), pp.2-es.

[5] Ali Rahimi, and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In Advances in neural information processing systems, pp. 1313–1320, 2008.

[6] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In Proc. 16th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '10, pages 1019–1028, New York, NY, USA, 2010. ACM.

[7] Seth A. Myers and Jure Leskovec. On the convexity of latent social network inference. In Proc. Neural Information Processing Systems (NIPS), 2010.

[8] Manuel Gomez Rodriguez, David Balduzzi, and Bernhard Scholkopf. Uncovering the temporal ¨ dynamics of diffusion networks. arXiv preprint arXiv:1105.0697, 2011.

[9] Maria-Florina Balcan and Nicholas J.A. Harvey. Learning submodular functions. In STOC, 2011.

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

[11] Nan Du, Le Song, Manuel Gomez Rodriguez, and Hongyuan Zha. Scalable influence estimation in continuous time diffusion networks. In Advances in Neural Information Processing Systems 26, 2013

[12] Nan Du, Yingyu Liang, Maria-Florina Balcan, and Le Song. Influence function learning in information diffusion networks. In ICML, 2014.

[13] Harikrishna Narasimhan, David C. Parkes, and Yaron Singer. Learnability of Influence in Networks. In Proc. 27th NIPS, 3168–3176, 2015.

[14] Alexandre Belloni, Tengyuan Liang, Hariharan Narayanan, and Alexander Rakhlin. Escaping the local minima via simulated annealing: Optimization of approximately convex functions. In Proceedings of The 28th Conference on Learning Theory, pp. 240–265, 2015.

[15] Dimitris Kalimeris, Yaron Singer, Karthik Subbian, and Udi Weinsberg. Learning Diffusion using Hyperparameters. In Proc. 35th ICML, 2420–2428, 2018.