Learning Opinions in Social Networks

Vincent Conitzer [*1] **Debmalya Panigrahi** [*1] **Hanrui Zhang** [*1]

Abstract

We study the problem of learning opinions in social networks. The learner observes the states of some sample nodes from a social network, and tries to infer the states of other nodes, based on the structure of the network. We show that sample-efficient learning is impossible when the network exhibits strong noise, and give a polynomial-time algorithm for the problem with nearly optimal sample complexity when the network is sufficiently stable.

1. Introduction

Suppose we are a social media company. A new product is about to come out, and we would like to learn whether each individual user has heard about it or not (we will refer to this as the current *opinion* of the user). This information may be useful for further marketing of the product, or for other purposes. To achieve this goal, we decide to run a poll, by asking each user visiting our website a few questions (i.e., *inspecting* the user). We hope to inspect as few users as possible, because we would rather let them engage in other activities on the site. Moreover, since we have no control over which users will visit our website, the only thing we can decide is the time interval during which we run the poll, or equivalently, the rough number of users to inspect.

If our goal is to make sure that our estimates for individual users are uniformly accurate, without further knowledge about users, we would have to inspect almost all users. However, in our role as a social media company, we have access to the *social network* formed by our users. In particular, we know which users are likely to be affected by which other users. This enables us to infer the opinion of some users without inspecting them at all, as long as we know the opinions of certain other users. For example, if user u strongly affects user v, and we know that user u is aware of our product, then we are quite sure that user v is also aware, because most likely u has told v about the product. In an extreme case, suppose we know that all users share all information with each other at all times. Then inspecting only one user is enough for our purpose, because either everyone is aware of the new product, or no one is. In general, how many users we need to inspect depends heavily on the structure of the network formed by the users. So, given the structure of the network, we want to know:

- How many users do we need to inspect in order to have enough information to make an accurate estimate of the current state? In other words, what is the *sample complexity* of learning opinions in a given network?
- Given the opinions of the users inspected, how can we infer the opinions of other users who have not been inspected?

While we have illustrated the problem using the example of marketing a product, the same model can be used for many other purposes. For instance, we may want to learn whether users are aware of a particular political candidate, or of a particular news item. We may want to learn whether an HIV awareness campaign has reached individual homeless youth (Wilder et al., 2018). Or, rather than the spread of information, we may consider the spread of a biological or computer virus in a network, learning where it is likely to have spread (Romano et al., 2010).

1.1. Our Results

In this paper, we give (1) a nearly tight bound on the sample complexity of learning opinions in social networks, and (2) a polynomial-time algorithm that outputs an approximately correct estimate of the state of the network with high probability. Given a desired error rate ε and failure probability δ, our goal is to design a learning algorithm that outputs, with probability at least $1-\delta$, an estimate of the state of the network that is accurate for at least $1-\varepsilon$ fraction of the nodes. We call this an (ε, δ)-learning algorithm. The sample complexity of such an algorithm is the number of labeled nodes (training samples) that it observes; we obtain an algorithm with nearly optimal sample complexity. Our results are summarized in the next theorem.

[*]Alphabetical order [1]Department of Computer Science, Duke University, Durham, USA. Correspondence to: Vincent Conitzer <conitzer@cs.duke.edu>, Debmalya Panigrahi <debmalya@cs.duke.edu>, Hanrui Zhang <hrzhang@cs.duke.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

Theorem 1.1 (Main Result, Informal)**.** *Suppose we are given a (possibly random) network $\mathcal{G}$ where information propagates by reachability, with an (expected) complexity parameter d. (We will define this parameter later.) Suppose also that $\varepsilon \in [\varepsilon_0, 1]$ and $\delta \in [\delta_0, 1]$ are the desired error rate and failure probability. (Here, ε_0 and δ_0 are network-dependent constants that we will define later.) Then, there exists an efficient (ε, δ)-learning algorithm for $\mathcal{G}$ that has a sample complexity of $m = \widetilde{O}(d/\varepsilon)$.[1] Moreover, the above number of samples is the minimum possible, up to a poly-logarithmic factor.*

1.2. Related Work

Most closely related to our work is the line of research on learning structures of social networks. The problem of interest there is a sort of inverse problem of the one studied in this paper: given outcomes of some propagation procedure, the goal is to recover parameters of the network governing the propagation. Some representative results in this domain include (Liben-Nowell & Kleinberg, 2007; Goyal et al., 2010; Chierichetti et al., 2011; Gomez Rodriguez et al., 2011; Saito et al., 2011; Du et al., 2012; Guille & Hacid, 2012; Abrahao et al., 2013; Cheng et al., 2014; Daneshmand et al., 2014; Du et al., 2014; Narasimhan et al., 2015; He et al., 2016; Kalimeris et al., 2018). A similar and more recent line of work is on representation learning for information propagation (Bourigault et al., 2016; Li et al., 2017; Wang et al., 2017). By virtue of being an inverse problem, our results are not directly comparable to all these.

The study of information propagation in a network was initiated by Kempe et al. (2003). Since then, various models of information propagation have been proposed (Gruhl et al., 2004; Chen et al., 2010; 2011; Myers et al., 2012). Several research questions have drawn significant attention in the context of information propagation, such as influence maximization (Mossel & Roch, 2007; Chen et al., 2009; 2010; 2011; Borgs et al., 2014; Tang et al., 2014), identification of influential nodes (Agarwal et al., 2008; Pal & Counts, 2011), and community detection (Faloutsos et al., 2004; Coscia et al., 2011).

Since the introduction of probably approximately correct (PAC) learning by Valiant (1984), a series of remarkable results have provided a rather complete picture for passive learning from observations. Following the groundbreaking Vapnik-Chervonenkis (VC) theory (Vapnik, 2013), various measures of complexity have been considered (Alon et al., 1997; Bartlett & Mendelson, 2002; Pollard, 2012; Daniely et al., 2015), based on which tighter generalization bounds have also been developed (Hanneke, 2016). While these general results are powerful, as discussed in later sections, they cannot be directly applied to the specific problem considered in this paper.

[1] $\widetilde{O}$ hides a poly-logarithmic factor.

2. Preliminaries

In this section, we review relevant concepts from learning theory and social network analysis, and formally define the problem investigated in this paper. Throughout the paper, for any set S and point x, let $S(x) = \mathbb{I}[x \in S]$ denote the indicator that $x \in S$.

2.1. The Theory of PAC Learning

The problem studied in this paper is an extension to the classical problem of *probably approximately correct (PAC) learning* (Valiant, 1984). In this problem, there is a space X of data points, a distribution $\mathcal{D}$ over X, and a hypothesis class $\mathcal{H} \subseteq 2^X$. We are also given the parameters $\epsilon, \delta > 0$, respectively denoting the desired error rate and failure probability. Now, given m iid samples $\{(x_i, y_i)\}_{i \in [m]}$ where $x_i \sim \mathcal{D}$ and $y_i = c(x_i)$ (recall that $c(x_i) = \mathbb{I}[x_i \in c]$) that correspond to a *ground truth* $c \in \mathcal{H}$, the goal of the (ε, δ) PAC learning algorithm is to return a hypothesis $h \in H$ such that with probability at least $1 - \delta$,

$$\Pr_{x \sim \mathcal{D}}[c(x) \neq h(x)] \leq \varepsilon.$$

We are interested in minimizing the number of samples m, i.e., the *sample complexity* of the algorithm. This is closely related to the *VC dimension* of the hypothesis class $\mathcal{H}$:

Definition 2.1 (VC Dimension)**.** A set S is *shattered* by a family of sets $\mathcal{F}$, if for any $T \subseteq S$, there exists $U \in \mathcal{F}$, such that $S \cap U = T$. The VC dimension of a hypothesis class $\mathcal{H}$ over a space X, denoted $\mathrm{VC}(\mathcal{H})$, is the cardinality of the largest set $S \subseteq X$ shattered by $\mathcal{H}$.

The sample complexity of PAC learning is given by the following theorem (see, e.g., (Kearns & Vazirani, 1994)):

Theorem 2.1 (VC Theorem, the Realizable Case)**.** *Fix X and $\mathcal{H}$. Consider any $\mathcal{D}$ over X, $c \in \mathcal{H}$, $\delta > 0$, and $\varepsilon > 0$. Now, given $m = O((\mathrm{VC}(\mathcal{H}) \log(1/\varepsilon) + \log(1/\delta))/\varepsilon)$ i.i.d. samples from D, the following holds with probability at least $1 - \delta$: any hypothesis $h \in H$ that is consistent with all the m samples (i.e., $h(x_i) = y_i$ for all $i \in [m]$) satisfies*

$$\Pr_{x \sim \mathcal{D}}[h(x) \neq c(x)] \leq \varepsilon.$$

Moreover, this bound is tight in the sense that any algorithm achieving this guarantee requires $\Omega((\mathrm{VC}(\mathcal{H}) + \log(1/\delta))/\varepsilon)$ samples.

2.2. Social Network Analysis

We now review a few basic concepts in social network analysis, which provide the language for describing how nodes of

the network interact with each other. These interactions enable inference of the opinions of nodes that are not inspected. Each node has two possible opinions or states, *active* and *inactive*, encoding whether the node is aware of an idea (e.g., a new product) or not. Following conventions in social network analysis, we assume the final opinions of nodes are formed via the following information propagation process: A subset of nodes (the *seed set*) is active at the beginning. Over time, active nodes make other (previously inactive) nodes active, according to some *propagation model*. Once a node becomes active, it never returns to the inactive state. The propagation stops eventually with a set of active nodes, which represents the final state of the network.

We consider a general propagation model based on *live-edge graphs* (see, e.g., (Chen et al., 2010)).[2] In this model, a network over nodes V (where $|V| = n$) is modeled by a distribution $\mathcal{G}$ over possible realizations of the network, each of which is a simple directed graph over V. Given the seed set $S_0 \subseteq V$, the propagation is governed by the following (random) process: First, a realized graph $G = (V, E) \sim \mathcal{G}$ is drawn from the distribution $\mathcal{G}$. Information then propagates in steps, where in each step, any previously inactive node v that has an edge (u, v) from a previously active node u becomes active. The set of active nodes in step i is given by:

$$S_i = S_{i-1} \cup \{u \mid \exists v \in S_{i-1},\, s.t.\, (v, u) \in E\}.$$

The above procedure defines a monotone sequence of sets of active nodes $S_0 \subseteq S_1 \subseteq S_2 \subseteq \ldots$. The propagation stops once $S_i = S_{i-1}$ for some $i > 0$, and this must happen at some stage since $|S_i| \leq n$ for all i. (In fact, for any $i \geq n - 1$, $S_i = S_{i+1}$.) We say $S_\infty = S_{n-1}$ is the final state of the network. Note that S_∞ is the set of nodes reachable from the seed set in the random graph G. (Recall that for any set of nodes $S \subseteq V$ and any node $u \in V$, $S(u) = \mathbb{I}[u \in S]$.)

Note that although we described the propagation model above as a deterministic process defined over a randomly generated network, it also captures the complementary model of random propagation in a fixed network. For instance, suppose we have a fixed network G_f, and for every edge (u, v), we are given a probability p_{uv} of node u activating node v along edge (u, v) (independent of other edges). To model this in our framework, we would first generate a random network G from the given network G_f, where edge (u, v) in G_f is realized independently with probability p_{uv} in G. Once this network is generated, we can define the propagation process as deterministic propagation in G based on reachability as defined above. Clearly, these are equivalent views and result in the same final network state.

[2] As far as we know, the live-edge graph model considered in this paper was developed in a folklore fashion. The name appeared in (Kempe et al., 2003), but there the actual model is more restricted, and is now often referred to as the triggering model.

2.3. Learning Opinions in Social Networks

We now proceed to define our problem formally. A problem instance comprises a set of nodes V (where $|V| = n$) and a (possibly random) network $\mathcal{G}$ over V (in the form of a live-edge graph defined above). The node set V is explicitly part of the input. The random network $\mathcal{G}$ (i.e., the corresponding probability distribution) may not be explicitly defined; indeed, its support can be exponentially large in the number of vertices. Nevertheless, the algorithm can draw i.i.d. samples from $\mathcal{G}$; we will typically call each such graph a sampled graph. Now, the actual propagation happens on a graph $G^* \sim \mathcal{G}$; we call this the realized graph. The algorithm does not have access to the realized graph.

Under these conditions, the goal is to design an algorithm that can "learn" S_∞ on the realized graph G^* for any seed set S_0 in the following sense: For any population distribution $\mathcal{D}$ defined on the set of vertices V, and for any values of parameters $\delta, \varepsilon > 0$, suppose the algorithm is given m i.i.d. labeled samples $\{(u_i, o_i)\}_{i \in [m]}$ where for any $i \in [m]$, $u_i \sim \mathcal{D}$ and

$$o_i = S_\infty(u_i) = \mathbb{I}[u_i \in S_\infty].$$

Then, the algorithm computes a hypothesis set $H \subseteq V$ of active nodes, such that with probability at least $1 - \delta$ over the random realized graph G^* from $\mathcal{G}$ and the random generation of the labeled samples from $\mathcal{D}$, the following holds:

$$\Pr_{u \sim \mathcal{D}}[S_\infty(u) \neq H(u)] \leq \varepsilon.$$

We are interested in minimizing the sample complexity m.

In words, given a network modeled by a live-edge graph, the problem asks for an algorithm, that, with high probability, approximately recovers the final outcome of a propagation process starting from any seed set, by observing i.i.d. samples of the outcome only. As described above, the algorithm is not aware of the realization of the network, i.e., the realized graph G^*, and this prevents us from applying existing results from learning theory directly.

Below is a concrete example of the problem of learning opinions in social networks. Again there is a fixed network G_f over nodes V, where every edge (u, v) is associated with a probability p_{uv} of node u activating node v along edge (u, v). Let the network $\mathcal{G}$ be the distribution of G in which every edge (u, v) realizes independently with probability p_{uv}. V and $\mathcal{G}$ constitute a problem instance — the algorithm knows V and can sample from $\mathcal{G}$. The procedure of learning with V and $\mathcal{G}$ then happens in the following way. First a seed set $S_0 \subseteq V$ and a population distribution $\mathcal{D}$ are chosen adversarially. Then nature draws a realized graph

$G^* \sim \mathcal{G}$ in which every edge (u, v) exists independently with probability p_{uv}, and propagation happens in G^* from S_0, resulting in S_∞. The algorithm then observes m i.i.d. labeled samples $\{(u_i, o_i)\}_{i\in[m]}$ where for any $i \in [m]$, $u_i \sim \mathcal{D}$ and $o_i = S_\infty(u_i)$. Based on these labeled samples, the algorithm computes a hypothesis $H \subseteq V$ which probably approximately recovers S_∞ with respect to $\mathcal{D}$.

We also remark that the problem becomes trivial with $\widetilde{\Omega}(n/\varepsilon)$ samples. In this case, one can learn the state of the network irrespective of (and ignoring) its structure. To this end, we aim to find a parameter of the network that tightly characterizes the minimum number of samples required to recover the outcome of the propagation.

3. Warmup: Deterministic Networks

To develop some intuition for the problem, we first investigate a special case where the network is deterministic.We present a few observations, given which the problem can be solved by applying the classical VC theorem.

Let $G = (V, E)$ be the only possible realization of $\mathcal{G}$. The key observation is that the effective hypothesis class to be considered might be much smaller than 2^V. In fact, for any $u \in V$ and $v \in V$ where v is reachable from u, for any seed set S_0, it is impossible that in the final outcome of the propagation, $u \in S_\infty$ but $v \notin S_\infty$. In general, we only need to consider hypotheses where no such contradictions happen. In other words, the propagation procedure on G induces a hypothesis class $\mathcal{H} \subseteq 2^V$. In light of the VC theorem, we now consider the VC dimension of this associated hypothesis class $\mathcal{H}$, which we define to be the VC dimension of the graph G. Projecting the definition of VC dimension to graphs, we get the following definition:

Definition 3.1 (VC Dimension of Directed Graphs)**.** The VC dimension $\mathrm{VC}(G)$ of a directed graph $G = (V, E)$ is the size of the maximum set $S \subseteq V$ of nodes, such that for any $u, v \in S$ where $u \neq v$, there is no u to v path in G.

In fact, the above definition coincides with the VC dimension of the hypothesis class associated with the graph.

Proposition 3.1. *Let G be any directed graph, and $\mathcal{H}$ be its associated hypothesis class defined above. Then* $\mathrm{VC}(G) = \mathrm{VC}(\mathcal{H})$.

Proof. Let $G = (V, E)$ be any directed graph, and $\mathcal{H}$ be its associated hypothesis class. We show below that a set of nodes $S \subseteq V$ is shattered by $\mathcal{H}$ if and only if there is no path from any node in S to any other node in S.

Suppose S satisfies for any $u, v \in S$ where $u \neq v$, there is no u to v path. Consider any subset of nodes $T \subseteq S$. We show that there exists some $S_0 \subseteq V$ such that the corresponding outcome of propagation $S_\infty \in \mathcal{H}$ satisfies $S_\infty \cap S = T$. In fact, let $S_0 = T$. Since for any $u \in T$ and $v \in S \setminus T$, u cannot reach v, we always have $(S \setminus T) \cap S_\infty = \emptyset$, which implies $S_\infty \cap S = T$.

Now suppose S is shattered by $\mathcal{H}$. Let $u, v \in S$ be any two nodes in S where $u \neq v$. We argue that there is no path from u to v. To see this, let $S_\infty \in \mathcal{H}$ be a feasible outcome of the propagation, where $u \in S_\infty$ and $v \notin S_\infty$. Such an outcome always exists, because S is shattered by $\mathcal{H}$. Then, if u can reach v, we have $u \in S_\infty \implies v \in S_\infty$, which is a contradiction. □

We make a few remarks regarding the above definition.

- Restricted to deterministic networks, one can always assume without loss of generality that the unique realization G is acyclic. This is because for any strongly connected component in G, either all nodes in the component are in S_∞, or none of the nodes is. One can therefore effectively contract any such component into a single node. Moreover, this contraction does not affect the VC dimension of G.
- Given the above observation, the definition of the VC dimension of a graph coincides precisely with the concept of the *width* of a graph (or that of a partially ordered set). As a result, the VC dimension of any graph can be computed in polynomial time (Felsner et al., 2003).
- Many natural graphs have small VC dimension. For example, a clique of any size, as well as a chain of any length, has VC dimension 1.

Based on the above observations, for deterministic $\mathcal{G}$, we may apply the classical VC theory to our problem as follows:

Theorem 3.1 (Learning Opinions in Social Networks, the Deterministic Case)**.** *Fix a deterministic network $G = (V, E)$. Consider any seed set S_0, distribution $\mathcal{D}$ over V, failure probability $\delta > 0$ and error rate $\varepsilon > 0$. Given $m = O((\mathrm{VC}(G)\log(1/\varepsilon) + \log(1/\delta))/\varepsilon)$ i.i.d. samples from $\mathcal{D}$, with probability at least $1 - \delta$, any hypothesis set $H \in \mathcal{H}$ in the associated hypothesis class that is consistent with all the samples satisfies*

$$\Pr_{u\sim\mathcal{D}}[S_\infty(u) \neq H(u)] \leq \varepsilon.$$

Moreover, this result is tight in that any algorithm providing this guarantee requires $\Omega((\mathrm{VC}(G)+\log(1/\delta))/\varepsilon)$ samples.

To compute a hypothesis H consistent with the samples, one can simply take all sampled nodes that are active (denoted A), and let H be the set of all nodes reachable from A in G. It is straightforward to check that H is in fact consistent with G and the sampled nodes. This gives an algorithm that is both sample efficient and computationally efficient.

4. The General Case: Random Networks

In the previous section, we demonstrated an interesting connection between the problem of learning opinions in social networks and the classical VC theory for passive learning. In particular, we identified a network dependent parameter, namely its width, that defines the VC dimension of the induced hypothesis class, and therefore dictates the sample complexity of learning on the network. We now generalize these results to random networks.

Now, the network $\mathcal{G}$ is no longer deterministic, but is a nontrivial probability distribution over graphs. This randomness of the network precludes the use of the VC theorem directly to obtain learning algorithms with a nontrivial sample complexity bound. In fact, the VC dimension $\mathrm{VC}(G)$ of the realized graph $G \sim \mathcal{G}$ is now random. At best, one may hope for a sample complexity bound that depends on the expectation of, or even perhaps some upper bound on, the value of $\mathrm{VC}(G)$ where $G \sim \mathcal{G}$. In this section, we first show that this, in general, is not possible. We then identify a condition under which sample efficient learning is possible, and give a nearly optimal learning algorithm under this condition.

4.1. Obstacles to Learning in General Networks

We first discuss a key difference between learning in deterministic and random networks, which makes it impossible to design sample efficient learning algorithms that work for *arbitrary* random networks. Recall that in this case, the learning algorithm does not know the actual realization of the random network. Since the labels of the sample nodes depend on both the seed set and the realized graph, the learning task becomes intuitively difficult when the randomness of the network "overwhelms" the information encoded in the seed set. We make this intuition concrete by showing that in certain pathological cases where $\mathcal{G}$ is poorly concentrated, any algorithm needs $\Omega(n)$ samples to learn the label of a single fixed node with constant probability, *even though* $\mathrm{VC}(G) = 1$ *for all realizable graphs* G *under* $\mathcal{G}$.

Consider the following construction of $\mathcal{G}$. To generate $G \sim \mathcal{G}$, we first generate a uniform random permutation $\sigma : V \to [n]$ of V, where $\sigma(u)$ is the rank of $u \in V$. $G = (V, E)$ is then defined by the following: for any $u, v \in V$,

$$(u, v) \in E \iff \sigma(u) + 1 = \sigma(v).$$

That is, G is a directed chain formed by connecting consecutive nodes in the permutation σ. Clearly G always has width 1, i.e., $\mathrm{VC}(G) = 1$.

Now fix the seed set S_0 to be a fixed node u_0, and let $\mathcal{D}$ be the uniform distribution over V. Observe that in the final outcome of the propagation, for any node $u \neq u_0$,

$$u \in S_\infty \iff \sigma(u_0) \leq \sigma(u),$$

where $\sigma(\cdot)$ is the permutation defining G. We can show the following lower bound:

Proposition 4.1. *By observing i.i.d. labeled samples, for any* $u \neq u_0$*, an algorithm needs* $\Omega(n)$ *samples to recover* $S_\infty(u)$ *with probability* $9/10$*.*

Proof. Observe that the Bayesian optimal algorithm for recovering $S_\infty(u)$ is the following. If u is among the sampled nodes, then output its label. Otherwise, output the majority label of the sampled nodes. In the argument below, we further allow the algorithm the additional advantage that it always knows the true majority label, and outputs this true majority label when u is not among the sampled nodes. There are two possible ways of correctly recovering $S_\infty(u)$:

- u happens to be one of the labeled samples.
- u happens to have the true majority label.

Suppose we observe $m = cn$ samples for some $c > 0$. The probability that the first case happens is $1 - (1 - 1/n)^m \approx 1 - 1/e^c$. For the second case, recall that the algorithm always outputs the true majority label. Even then, the probability that the majority label coincides with $S_\infty(u)$ is given by

$$\sum_{n/2 \leq i \leq n} \frac{2}{n} \cdot \frac{i}{n} \approx \frac{3}{4}.$$

Taking a union bound over these two cases, the probability of the algorithm outputting the correct label is at most $\frac{7}{4} - \frac{1}{e^c}$, which is less than $9/10$ for small enough c. □

In other words, one needs to inspect a constant fraction of the nodes in order to obtain a constant error rate.

4.2. Stability of Networks

Proposition 4.1 rules out the possibility of any non-trivial sample complexity bounds without making further assumptions on the network. In the counterexample, the main obstacle to an efficient learning algorithm is the lack of concentration of $\mathcal{G}$, for any reasonable notion of concentration. To this end, we identify a mild condition that captures the intrinsic "stability" of random networks, and give a learning algorithm subject to this condition.

Definition 4.1 ($(\varepsilon_0, \delta_0)$-Stable Networks)**.** Consider a network $\mathcal{G}$ with node set V, and let G and G' be two i.i.d. graphs drawn from $\mathcal{G}$. Then, $\mathcal{G}$ is said to be $(\varepsilon_0, \delta_0)$-stable with respect to a distribution $\mathcal{D}$ defined over V, if the final outcomes of propagation S_∞ and S'_∞ in G and G' respectively from any seed set S_0, satisfy the following: with probability at least $1 - \delta_0$ over G,

$$\Pr_{G' \sim \mathcal{G}, u \sim \mathcal{D}}[S_\infty(u) \neq S'_\infty(u)] \leq \varepsilon_0.$$

Algorithm 1 Learning Algorithm for General Networks

Input: A network $\mathcal{G}$ over V that is $(\varepsilon_0, \delta_0)$-stable, m labeled sample nodes $\{(u_i, o_i)\}_{i\in[m]}$, desired error rate $\varepsilon \geq C_1 \cdot \varepsilon_0$ and failure probability $\delta \geq C_2 \cdot \delta_0$. ($C_1$ and C_2 are absolute constants.)
Output: A hypothesis set $H \subset V$.

Draw $\theta := \frac{100}{\varepsilon} \log \frac{1}{\delta}$ i.i.d. sample graphs $\{G^{(k)}\}_{k\in[\theta]}$, where $G^{(k)} \sim \mathcal{G}$ for all $k \in [\theta]$.
for $k \in [\theta]$ **do**
  Compute hypothesis set $H^{(k)}$ by calling Algorithm 2 on $G^{(k)}$ and $\{(u_i, o_i)\}_{i\in[m]}$.
end for
$H \leftarrow \emptyset$.
for $u \in V$ **do**
  $c_u = \sum_{k\in[\theta]} H^{(k)}(u)$.
  $H \leftarrow H \cup \{u\}$ if $c_u \geq \theta/2$.
end for
Return H.

Algorithm 2 Empirical Risk Minimization in Networks

Input: A graph $G = (V, E)$, m labeled sample nodes $\{(u_i, o_i)\}_{i\in[m]}$.
Output: A hypothesis set $H \subseteq V$.

Create a capacitated graph $G' = (V', E')$. Let $V' \leftarrow V \cup \{s, t\}$, $E' \leftarrow E$. Assign all edges currently in E' capacity ∞.
for $i \in [m]$ **do**
  if $o_i = 1$ **then**
    Let $E' \leftarrow E' \cup \{(s, u_i)\}$, where the new edge (s, u_i) has capacity 1.
  else
    Let $E' \leftarrow E' \cup \{(u_i, t)\}$, where the new edge (u_i, t) has capacity 1.
  end if
end for
Compute an s-t min-cut, let $H \subseteq V$ be the set of nodes of G on the s-side of the cut. Return H.

In words, a network is $(\varepsilon_0, \delta_0)$-stable if the outcome S_∞ of the propagation from S_0 does not lie in the "tail" with probability at least $1 - \delta_0$, in which case the expected distance from the outcome S_∞ to an i.i.d. copy of itself S'_∞ is at most ε_0. This is rather mild, since it does not even require the expected distance between two independent outcomes of propagation to have small second moment, even conditioning on one of them being not in the tail. Technically, with a small second moment, one would expect good concentration behavior of S_∞, making it practically deterministic. This is not the case for the notion of $(\varepsilon_0, \delta_0)$-stability. As we will see, our algorithm is independent of any explicit concentration property of S_∞.

We also remark that the notion of $(\varepsilon_0, \delta_0)$-stability may not be the only nontrivial condition that permits efficient learning. On the other hand, it does provide a way of parameterizing random networks which allows us to derive almost matching upper and lower bounds on the sample complexity. To this end, $(\varepsilon_0, \delta_0)$-stability appears to be a natural notion that captures the intrinsic resolution of random networks, and may generalize to other learning settings.

4.3. Efficient Algorithm for Stable Networks

Our learning algorithm, Algorithm 1, works in the following way. First, the algorithm draws a number of i.i.d. sample graphs from $\mathcal{G}$ (note that it does not have access to the realized graph in which the propagation happens; if it did, we would be in the deterministic case given in the previous section). The algorithm then tries to fit the labels of the sampled nodes for each of these sampled graphs separately by finding the hypothesis set that is consistent (i.e., minimizing the number of sampled nodes whose labels are inconsistent with the hypothesis). In other words, the algorithm finds the *empirical risk minimizer* (ERM) for each of the sampled graphs. Note that here, the best hypothesis set for a sampled graph may not be perfectly consistent with the sampled nodes — this is because the sampled nodes are part of the outcome of the propagation process on the actual realization of the network, which is possibly different from the sampled graphs. With these empirical risk minimizers, the algorithm then computes and outputs the node-wise majority vote, i.e., a node is in the output hypothesis if and only if it is in more than half of these ERMs.

To efficiently compute ERMs on sampled graphs, Algorithm 1 calls a subroutine Algorithm 2. Algorithm 2 treats the ERM problem as a constrained combinatorial optimization problem. Concretely, it models the problem in the following way: each sample node with label 1 has weight 1, and each sample node with label 0 has weight -1. The algorithm tries to find a subset of all nodes with maximum weight, subject to the constraint that for any edge $(u, v) \in E$, if u is in the subset, then v must also be in the subset. This problem turns out to be polynomial-time solvable via a minimum cut subroutine.

We now analyze the efficiency and correctness of Algorithm 1.

Theorem 4.1 (Learning Opinions in Social Networks, the General Case)**.** *Let $\mathcal{G}$ be a (possibly random) network defined on a node set V that is $(\varepsilon_0, \delta_0)$-stable with respect to some distribution $\mathcal{D}$ over V. Let $S_0 \subseteq V$ be any seed set.*

Further, suppose Algorithm 1 is given

$$m = O\left(\frac{\mathbb{E}_{G\sim\mathcal{G}}[\mathrm{VC}(G)]\log(1/\varepsilon)+\log(1/\delta)}{\varepsilon}\right)$$

random samples, where $\delta \geq C_1 \cdot \delta_0$ and $\varepsilon \geq C_2 \cdot \varepsilon_0$. Then, with probability at least $1-\delta$, Algorithm 1 outputs a hypothesis set $H \subseteq V$ that satisfies

$$\Pr_{u\sim\mathcal{D}}[S_\infty(u) \neq H(u)] \leq \varepsilon.$$

Moreover, Algorithm 1 runs in time $\mathrm{poly}(n, 1/\varepsilon, \log(1/\delta))$. *(Here, $C_1 > 0$ and $C_2 > 0$ above are absolute constants.)*

Before proving the theorem, we remark that the above sample complexity bound is optimal up to a factor of $O(\log(1/\varepsilon))$. This can be seen by restricting $\mathcal{G}$ to be deterministic, and comparing to the lower bound in Theorem 3.1.

Proof of Theorem 4.1. First observe that Algorithm 1 runs in time $\mathrm{poly}(n, 1/\varepsilon, \log(1/\delta))$. This is because the algorithm takes $m = \mathrm{poly}(n, 1/\varepsilon, \log(1/\delta))$ labeled sample nodes, and performs empirical risk minimization with these nodes on $\theta = \mathrm{poly}(n, 1/\varepsilon, \log(1/\delta))$ sampled graphs. In each of these, the algorithm calls Algorithm 2 on an input of polynomial size. The latter simply computes a min-cut which is polynomial-time in the input size, and therefore, is $\mathrm{poly}(n, 1/\varepsilon, \log(1/\delta))$. Finally, the algorithm scans through the ERMs and the nodes, and computes the majority vote in polynomial time.

Now we show Algorithm 1 in fact outputs a hypothesis with the desired error rate and failure probability. We first show that Algorithm 2 is correct, i.e., it does return an ERM:

Lemma 4.1. *Given a graph $G = (V, E)$ and labeled sample nodes $\{(u_i, o_i)\}_{i\in[m]}$, Algorithm 2 finds a set $H \subseteq V$ consistent with G that minimizes the empirical risk given by:*

$$\sum_{i\in[m]} \mathbb{I}[H(u_i) \neq o_i].$$

Proof. Let $w : V \to \mathbb{Z}$ be a weight function over nodes. We construct w such that for any $u \in V$,

$$w(u) = \sum_{i\in[m]} \mathbb{I}[u = u_i] \cdot (2o_i - 1).$$

Observe that minimizing the empirical risk is equivalent to finding a set $H \subseteq V$ consistent with G that maximizes the total weight $\sum_{u\in H} w(u)$. In fact,

$$\begin{aligned}
\sum_{u\in H} w(u) &= \sum_{u\in H}\sum_{i\in[m]} \mathbb{I}[u = u_i] \cdot (2o_i - 1) \\
&= \sum_{i\in[m]} (2o_i - 1) \sum_{u\in H} \mathbb{I}[u = u_i] \\
&= \sum_{i\in[m]} (2o_i - 1) \cdot H(u_i) \\
&= \sum_{i\in[m]} (o_i - \mathbb{I}[H(u_i) \neq o_i]) \\
&= \sum_{i\in[m]} o_i - \sum_{i\in[m]} \mathbb{I}[H(u_i) \neq o_i].
\end{aligned}$$

We further argue that maximizing $\sum_{u\in H} w(u)$ is equivalent to the min-cut in Algorithm 2. Observe that in any finite capacity s-t cut in G', for any edge $e = (u, v) \in E$ in the graph G, it cannot be the case that u is in the s-side of the cut, and v is in the t-side of the cut. As a result, finite capacity s-t cuts in G' correspond to feasible outcomes of propagation in G. In fact, fixing a finite capacity cut (where all nodes in the s-side of the cut form a set H), for each node $u \in V$ where $w(u) \neq 0$, there are two cases:

- u is in the s-side of the cut, or equivalently, $u \in H$. When this happens, we have to cut all edges from u to t, each of which corresponds to a sampled node (u_i, o_i) where $u_i = u$ and $o_i = 0$. In this case, we incur cost
$$\sum_{i\in[m]} \mathbb{I}[u_i = u](1 - o_i) = \sum_{i\in[m]} \mathbb{I}[u_i = u](H(u) - o_i).$$
- u is in the t-side of the cut, or equivalently, $u \notin H$. When this happens, we have to cut all edges from s to u, each of which corresponds to a sampled node (u_i, o_i) where $u_i = u$ and $o_i = 1$. In this case, we incur cost
$$\sum_{i\in[m]} \mathbb{I}[u_i = u](o_i - 0) = \sum_{i\in[m]} \mathbb{I}[u_i = u](o_i - H(u)).$$

So the total cost we incur can be written as

$$\begin{aligned}
&\sum_{u\in V}\sum_{i\in[m]} \mathbb{I}[u_i = u](H(u) - o_i) \cdot (2H(u) - 1) \\
&= \sum_{u\in V}\sum_{i\in[m]} \mathbb{I}[u_i = u](2H(u)^2 - H(u) - 2H(u)o_i + o_i) \\
&= \sum_{u\in V}\sum_{i\in[m]} \mathbb{I}[u_i = u](H(u)(1 - 2o_i) + o_i) \\
&= \sum_{u\in H} -w(u) + \sum_{i\in[m]} o_i.
\end{aligned}$$

Observe that the min-cut minimizes the above cost, which is equivalent to maximizing $\sum_{u\in H} w(u)$. This concludes the proof of the lemma. □

We can now apply the following lemma about the error rate of ERMs:

Lemma 4.2. *Fix a feature space X and a hypothesis class $\mathcal{H} \subseteq 2^X$. Fix any distribution $\mathcal{D}$ over X, $c \subseteq X$, $\delta' > 0$ and $\varepsilon' > 0$. Moreover, suppose there is a hypothesis $h^* \in \mathcal{H}$ such that $\Pr_{x\sim\mathcal{D}}[h^*(x) \neq c(x)] \leq \varepsilon'/2$. Now, consider any ERM $h \in \mathcal{H}$ for at least $m' = O((\mathrm{VC}(\mathcal{H})\log(1/\varepsilon') + \log(1/\delta'))/\varepsilon')$ samples $\{(x_i, y_i)\}_{i\in[m']}$, i.e.,*

$$h \in \mathrm{argmin}_{h'\in\mathcal{H}} \sum_{i\in[m]} \mathbb{I}[h'(x_i) \neq y_i].$$

Then, with probability at least $1-\delta'$, h has error rate at most ε', i.e., $\Pr_{x\sim\mathcal{D}}[h(x) \neq c(x)] \leq \varepsilon'$.

The lemma is a straightforward adaptation of the classical VC theorem, and can be proved by modifying the original proof, replacing the additive concentration bounds by their multiplicative versions.

We now bound the error rate and failure probability of Algorithm 1. Let S_∞ be the outcome of the actual propagation from S_0 on the realized graph G^*. Moreover, for any $k \in [\theta]$, let $S_\infty^{(k)}$ be the outcome of the propagation from S_0 on the sampled graph $G^{(k)}$. Since $\mathcal{G}$ is $(\varepsilon_0, \delta_0)$-stable, with probability at least $1-\delta_0$ over the random choice of G^*, the following holds:

$$\text{for any } k \in [\theta], \Pr_{G^{(k)}\sim\mathcal{G}, u\sim\mathcal{D}}[S_\infty(u) \neq S_\infty^{(k)}(u)] \leq \varepsilon_0.$$

We condition on the above event from now on. Otherwise, we consider the algorithm to have failed (which happens with probability at most δ_0).

For $k \in [\theta]$, let $e^{(k)}$ be the difference between S_∞ and $S_\infty^{(k)}$:

$$e^{(k)} = \Pr_{u\sim\mathcal{D}}[S_\infty(u) \neq S_\infty^{(k)}(u)].$$

Observe that $\{e^{(k)}\}_{k\in[\theta]}$ are i.i.d. random variables in $[0,1]$, whose expectation does not exceed ε_0.

We now apply Lemma 4.2 with different parameters (particularly, different $\varepsilon' = \varepsilon^{(k)}$) for each $G^{(k)}$, to bound the error rate of $H^{(k)}$ w.r.t. the actual outcome S_∞. For any $k \in [\theta]$, apply Lemma 4.2 with $m' = m$, $\delta' = \delta/3\theta$ and

$$\varepsilon' = \varepsilon^{(k)} = \max\left(2e^{(k)}, \frac{\varepsilon}{8}\cdot\left(1 + \frac{\mathrm{VC}(G^{(k)})}{\mathbb{E}_{G\sim\mathcal{G}}[\mathrm{VC}(G)]}\right)\right).$$

Note that the condition of Lemma 4.2 is satisfied. In particular, there is a hypothesis $S_\infty^{(k)}$ consistent with $G^{(k)}$ that has error rate $e^{(k)} \leq \varepsilon^{(k)}/2$. Then,

$$\begin{aligned}
&m' \\
&= O\left(\frac{\mathrm{VC}(G^{(k)})\log(1/\varepsilon') + \log(1/\delta')}{\varepsilon'}\right) \\
&\leq O\left(\frac{((8+o(1))(\mathrm{VC}(G^{(k)})\log(1/\varepsilon) + \log(1/\delta))}{\varepsilon(1 + \mathrm{VC}(G^{(k)})/\mathbb{E}_{G\sim\mathcal{G}}[\mathrm{VC}(G)])}\right) \\
&\leq O\left(\frac{(8+o(1))(\mathbb{E}_{G\sim\mathcal{G}}[\mathrm{VC}(G)]\log(1/\varepsilon) + \log(1/\delta))}{\varepsilon}\right) \\
&\leq m,
\end{aligned}$$

where the last inequality holds when the constant in the choice of m is large enough. So, by Lemma 4.2, we get the following: with probability at least $1-\delta'$, $H^{(k)}$ satisfies

$$\Pr_{u\sim\mathcal{D}}[S_\infty(u) \neq H^{(k)}(u)] \leq \varepsilon^{(k)}.$$

Taking a union bound over $k \in [\theta]$, this inequality holds simultaneously for all $H^{(k)}$ with probability at least $1 - \theta\cdot\delta' \geq 1 - \delta/3$. Again, we condition on the above event from now on, and consider the algorithm to have failed otherwise (which happens with probability at most $\delta/3$).

Observe that the expected error rate of $\{H^{(k)}\}_k$ is already low (i.e., on the order of ε). To be specific, note that $\{G^{(k)}\}_k$ are still i.i.d. variables with distribution $\mathcal{G}$ even if we condition on the event that Algorithm 2 succeeds for all $\{G^k\}_k$, since the latter depends only on the randomness in the labeled sample nodes. As a result, for any $k \in [\theta]$, we have:

$$\begin{aligned}
&\mathbb{E}_{G^{(k)}}[\varepsilon^{(k)}] \\
&= \mathbb{E}_{G^{(k)}}\left[\max\left(2e^{(k)}, \frac{\varepsilon}{8}\left(1 + \frac{\mathrm{VC}(G^{(k)})}{\mathbb{E}_{G\sim\mathcal{G}}[\mathrm{VC}(G)]}\right)\right)\right] \\
&\leq \mathbb{E}_{G^{(k)}}\left[2e^{(k)} + \frac{\varepsilon}{8}\left(1 + \frac{\mathrm{VC}(G^{(k)})}{\mathbb{E}_{G\sim\mathcal{G}}[\mathrm{VC}(G)]}\right)\right] \\
&\leq 2\mathbb{E}_{G^{(k)}}[e^{(k)}] + \frac{\varepsilon}{8} + \varepsilon\cdot\frac{\mathbb{E}_{G^{(k)}}[\mathrm{VC}(G^{(k)})]}{8\mathbb{E}_{G\sim\mathcal{G}}[\mathrm{VC}(G)]} \\
&\leq 2\varepsilon_0 + \frac{\varepsilon}{4}.
\end{aligned}$$

Since $\varepsilon \geq C_1\cdot\varepsilon_0$, the above is upper bounded by $\varepsilon/3$ whenever $C_1 \geq 24$. But, each $H^{(k)}$ may still have error rate larger than ε with probability larger than δ. Here, we apply majority voting to boost the probability of success.

First, we bound the average error rate of $\{H^{(k)}\}_k$ using concentration inequalities, and show that with high probability (i.e., at least $1-\delta/3$), it is at most $\varepsilon/2$. Observe that $\{\varepsilon^{(k)}\}_k$ are i.i.d. variables in $[0,1]$.[3] For small enough δ, by

[3] The above choice of ε^k itself may exceed 1. However, since $\varepsilon^{(k)}$ is an upper bound of a probability, one can always truncate $\varepsilon^{(k)}$ at 1, which does not increase the mean. We omit this in the proof for the sake of brevity.

the multiplicative Chernoff bound,

$$\begin{aligned}
&\Pr\left[\frac{1}{\theta}\sum_{k\in[\theta]}\varepsilon^{(k)} \geq \frac{\varepsilon}{2}\right] \\
&= \Pr\left[\frac{1}{\theta}\sum_{k\in[\theta]}\varepsilon^{(k)} \geq \left(1+\frac{1}{2}\right)\cdot\frac{\varepsilon}{3}\right] \\
&\leq \exp\left(-\frac{(1/2)^2\cdot(\theta\varepsilon/3)}{2+1/2}\right) \\
&= \exp\left(\frac{10\log(\delta)}{3}\right) \leq \frac{\delta}{3}.
\end{aligned}$$

We remark that the actual mean of $\varepsilon^{(k)}$ may be smaller than $\varepsilon/3$, but that only makes the probability smaller. So with probability at least $1-\frac{\delta}{3}$,

$$\frac{1}{\theta}\sum_{k\in[\theta]}\varepsilon^{(k)} \leq \frac{\varepsilon}{2}.$$

The final step is to show that the majority vote amplifies the average error rate of the ERMs by at most a factor of 2.

Lemma 4.3. *Fix a feature space X, a distribution $\mathcal{D}$ over X, and $c \subseteq X$. Suppose there are θ subsets of X, $\{h^{(k)}\}_{k\in[\theta]}$, satisfying*

$$\frac{1}{\theta}\sum_{k\in[\theta]}\Pr_{x\sim\mathcal{D}}[c(x)\neq h^{(k)}(x)] \leq \varepsilon',$$

for some $\varepsilon' > 0$. Then the pointwise majority vote h of $\{h^{(k)}\}_{k\in[\theta]}$, defined such that for any $x \in X$,

$$h(x) = \mathbb{I}\left[\sum_{k\in[\theta]} h^{(k)}(x) \geq \frac{\theta}{2}\right],$$

satisfies $\Pr_{x\sim\mathcal{D}}[h(x)\neq c(x)] \leq 2\varepsilon'$.

Proof. Fix some $x \in X$. Suppose $c(x) \neq h(x)$. Then it has to be the case that

$$\sum_{k\in[\theta]}\mathbb{I}[h^{(k)}(x)\neq c(x)] \geq \frac{\theta}{2}.$$

So it always holds that

$$\frac{1}{\theta}\sum_{k}\mathbb{I}[h^{(k)}(x)\neq c(x)] \geq \frac{1}{2}\cdot\mathbb{I}[h(x)\neq c(x)].$$

Now one may bound the difference between h and c in the following way.

$$\begin{aligned}
\Pr_{x\sim\mathcal{D}}[h(x)\neq c(x)] &= \mathbb{E}_{x\sim\mathcal{D}}[\mathbb{I}[h(x)\neq c(x)]] \\
&\leq \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{2}{\theta}\sum_{k\in[\theta]}\mathbb{I}[h^{(k)}(x)\neq c(x)]\right] \\
&= \frac{2}{\theta}\sum_{k\in\theta}\mathbb{E}_{x\sim\mathcal{D}}[\mathbb{I}[h^{(k)}(x)\neq c(x)]] \\
&= \frac{2}{\theta}\sum_{k\in\theta}\Pr_{x\sim\mathcal{D}}[h^{(k)}(x)\neq c(x)] \\
&\leq 2\varepsilon',
\end{aligned}$$

which gives the desired bound. □

Now we apply Lemma 4.3 to $\{H^{(k)}\}_{k\in[\theta]}$, with $\varepsilon' = \varepsilon/2$. The condition is satisfied, since the error rate of $H^{(k)}$ is upper bounded by $\varepsilon^{(k)}$, and the average of these $\{\varepsilon^{(k)}\}_k$ is at most $\varepsilon/2$. As a result, the majority vote H, which is the output of Algorithm 1, has error rate at most ε. As for the failure probability, recall that the algorithm may fail in 3 cases: (1) the realized graph G^* lies in the tail, so no property can be guaranteed by the (ε_0,δ_0)-stability of $\mathcal{G}$, which happens with probability at most δ_0, (2) one of the calls to Algorithm 2 fails, which happens with probability at most $\delta/3$ over the labeled sample nodes, and (3) the upper bound on the average error rate of $\{H^{(k)}\}_k$ exceeds $\varepsilon/2$, which happens with probability at most $\delta/3$ over the sampled graphs $\{G^{(k)}\}_k$. So, taking a union bound over these three cases, we infer that the total probability of failure does not exceed $\delta \geq 3\delta_0$ for any $C_2 \geq 3$. This concludes the proof of Theorem 4.1. □

5. Conclusion and Future Research

While various aspects of information propagation in social networks have been extensively studied, the problem of inferring the state of a network based on the structure induced by such propagation procedures has remained largely unexplored. In this paper, we study the algorithmic and statistical aspects of learning opinions in social networks. Our results show that in the classical live-edge graph model, nontrivial (and in fact, nearly optimal) inference is possible if and only if the noise exhibited by the network is not overwhelmingly large. Future research directions include generalizing our results to other models of social networks, as well as other notions of information propagation. Also, our results can be interpreted as a generalization of the VC theorem to random hypothesis classes with some specific properties. Another interesting research question is whether such generalization is possible for more general hypothesis classes. Answering these questions would broaden the understanding of both social network analysis and the theory of passive learning.

Acknowledgements

We thank Wei Chen, Shang-Hua Teng, and anonymous reviewers for helpful feedback. VC and HZ are supported by NSF award IIS-1814056. DP is supported in part by NSF awards CCF-1535972, CCF-1955703, an NSF CAREER Award CCF-1750140, and the Indo-US Center on Algorithms under Uncertainty.

References

Abrahao, B., Chierichetti, F., Kleinberg, R., and Panconesi, A. Trace complexity of network inference. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 491–499. ACM, 2013.

Agarwal, N., Liu, H., Tang, L., and Yu, P. S. Identifying the influential bloggers in a community. In *Proceedings of the 2008 international conference on web search and data mining*, pp. 207–218. ACM, 2008.

Alon, N., Ben-David, S., Cesa-Bianchi, N., and Haussler, D. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM (JACM)*, 44(4):615–631, 1997.

Bartlett, P. L. and Mendelson, S. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Borgs, C., Brautbar, M., Chayes, J., and Lucier, B. Maximizing social influence in nearly optimal time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 946–957. SIAM, 2014.

Bourigault, S., Lamprier, S., and Gallinari, P. Representation learning for information diffusion through social networks: an embedded cascade model. In *Proceedings of the Ninth ACM international conference on Web Search and Data Mining*, pp. 573–582. ACM, 2016.

Chen, W., Wang, Y., and Yang, S. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 199–208. ACM, 2009.

Chen, W., Wang, C., and Wang, Y. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1029–1038. ACM, 2010.

Chen, W., Collins, A., Cummings, R., Ke, T., Liu, Z., Rincon, D., Sun, X., Wang, Y., Wei, W., and Yuan, Y. Influence maximization in social networks when negative opinions may emerge and propagate. In *Proceedings of the 2011 siam international conference on data mining*, pp. 379–390. SIAM, 2011.

Cheng, J., Adamic, L., Dow, P. A., Kleinberg, J. M., and Leskovec, J. Can cascades be predicted? In *Proceedings of the 23rd international conference on World wide web*, pp. 925–936. ACM, 2014.

Chierichetti, F., Liben-nowell, D., and Kleinberg, J. M. Reconstructing patterns of information diffusion from incomplete observations. In *Advances in neural information processing systems*, pp. 792–800, 2011.

Coscia, M., Giannotti, F., and Pedreschi, D. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 4(5):512–546, 2011.

Daneshmand, H., Gomez-Rodriguez, M., Song, L., and Schölkopf, B. Estimating diffusion network structures: Recovery conditions, sample complexity & soft-thresholding algorithm. In *International Conference on Machine Learning*, pp. 793–801, 2014.

Daniely, A., Sabato, S., Ben-David, S., and Shalev-Shwartz, S. Multiclass learnability and the erm principle. *The Journal of Machine Learning Research*, 16(1):2377–2404, 2015.

Du, N., Song, L., Yuan, M., and Smola, A. J. Learning networks of heterogeneous influence. In *Advances in Neural Information Processing Systems*, pp. 2780–2788, 2012.

Du, N., Liang, Y., Balcan, M., and Song, L. Influence function learning in information diffusion networks. In *International Conference on Machine Learning*, pp. 2016–2024, 2014.

Faloutsos, C., McCurley, K. S., and Tomkins, A. Fast discovery of connection subgraphs. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 118–127. ACM, 2004.

Felsner, S., Raghavan, V., and Spinrad, J. Recognition algorithms for orders of small width and graphs of small dilworth number. *Order*, 20(4):351–364, 2003.

Gomez Rodriguez, M., Balduzzi, D., and Schölkopf, B. Uncovering the temporal dynamics of diffusion networks. In *28th International Conference on Machine Learning (ICML 2011)*, pp. 561–568. International Machine Learning Society, 2011.

Goyal, A., Bonchi, F., and Lakshmanan, L. V. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pp. 241–250. ACM, 2010.

Gruhl, D., Guha, R., Liben-Nowell, D., and Tomkins, A. Information diffusion through blogspace. In *Proceedings of the 13th international conference on World Wide Web*, pp. 491–501. ACM, 2004.

Guille, A. and Hacid, H. A predictive model for the temporal dynamics of information diffusion in online social networks. In *Proceedings of the 21st international conference on World Wide Web*, pp. 1145–1152. ACM, 2012.

Hanneke, S. The optimal sample complexity of pac learning. *The Journal of Machine Learning Research*, 17(1):1319–1333, 2016.

He, X., Xu, K., Kempe, D., and Liu, Y. Learning influence functions from incomplete observations. In *Advances in Neural Information Processing Systems*, pp. 2073–2081, 2016.

Kalimeris, D., Singer, Y., Subbian, K., and Weinsberg, U. Learning diffusion using hyperparameters. In *International Conference on Machine Learning*, pp. 2425–2433, 2018.

Kearns, M. J. and Vazirani, U. V. *An introduction to computational learning theory*. MIT press, 1994.

Kempe, D., Kleinberg, J., and Tardos, É. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146. ACM, 2003.

Li, C., Ma, J., Guo, X., and Mei, Q. Deepcas: An end-to-end predictor of information cascades. In *Proceedings of the 26th international conference on World Wide Web*, pp. 577–586. International World Wide Web Conferences Steering Committee, 2017.

Liben-Nowell, D. and Kleinberg, J. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7): 1019–1031, 2007.

Mossel, E. and Roch, S. On the submodularity of influence in social networks. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pp. 128–134. ACM, 2007.

Myers, S. A., Zhu, C., and Leskovec, J. Information diffusion and external influence in networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 33–41. ACM, 2012.

Narasimhan, H., Parkes, D. C., and Singer, Y. Learnability of influence in networks. In *Advances in Neural Information Processing Systems*, pp. 3186–3194, 2015.

Pal, A. and Counts, S. Identifying topical authorities in microblogs. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 45–54. ACM, 2011.

Pollard, D. *Convergence of stochastic processes*. Springer Science & Business Media, 2012.

Romano, C. M., de Carvalho-Mello, I. M. G., Jamal, L. F., de Melo, F. L., Iamarino, A., Motoki, M., Pinho, J. R. R., Holmes, E. C., de Andrade Zanotto, P. M., Consortium, V., et al. Social networks shape the transmission dynamics of hepatitis c virus. *PLoS One*, 5(6), 2010.

Saito, K., Ohara, K., Yamagishi, Y., Kimura, M., and Motoda, H. Learning diffusion probability based on node attributes in social networks. In *International Symposium on Methodologies for Intelligent Systems*, pp. 153–162. Springer, 2011.

Tang, Y., Xiao, X., and Shi, Y. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 75–86. ACM, 2014.

Valiant, L. G. A theory of the learnable. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pp. 436–445. ACM, 1984.

Vapnik, V. *The nature of statistical learning theory*. Springer science & business media, 2013.

Wang, J., Zheng, V. W., Liu, Z., and Chang, K. C.-C. Topological recurrent neural network for diffusion prediction. In *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 475–484. IEEE, 2017.

Wilder, B., Immorlica, N., Rice, E., and Tambe, M. Maximizing influence in an unknown social network. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.