Learning Diffusion using Hyperparameters

Dimitris Kalimeris [1] **Yaron Singer** [1] **Karthik Subbian** [2] **Udi Weinsberg** [2]

Abstract

In this paper we advocate for a hyperparametric approach to learn diffusion in the independent cascade (IC) model. The sample complexity of this model is a function of the number of edges in the network and consequently learning becomes infeasible when the network is large. We study a natural restriction of the hypothesis class using additional information available in order to dramatically reduce the sample complexity of the learning process. In particular we assume that diffusion probabilities can be described as a function of a global hyperparameter and features of the individuals in the network. One of the main challenges with this approach is that training a model reduces to optimizing a non-convex objective. Despite this obstacle, we can shrink the best-known sample complexity bound for learning IC by a factor of $|E|/d$ where $|E|$ is the number of edges in the graph and d is the dimension of the hyperparameter. We show that under mild assumptions about the distribution generating the samples one can provably train a model with low generalization error. Finally, we use large-scale diffusion data from Facebook to show that a hyperparametric model using approximately 20 features per node achieves remarkably high accuracy.

1. Introduction

For well over a decade there has been extensive work on learning social network influence models (Liben-Nowell & Kleinberg, 2003; Netrapalli & Sanghavi, 2012; Abrahao et al., 2013; Friggeri et al., 2014; Anderson et al., 2015; Subbian et al., 2017), and the independent cascade model in particular (Saito et al., 2008; Gomez Rodriguez et al., 2010; Goyal et al., 2010; Gomez Rodriguez et al., 2011; Du et al., 2014; Lemonnier et al., 2014; Bourigault et al., 2014; Narasimhan et al., 2015). Independent cascade (IC) was popularized by the seminal work of Kempe, Kleinberg, and Tardos (Kempe et al., 2003) and is a stochastic model that predicts the likelihood of information diffusing from one individual to another in a social network. In this model for every pair of individuals connected in the network u, v there is a probability $p_{u,v}$ that v adopts the behavior of u (i.e. information is *diffused* from u to v).

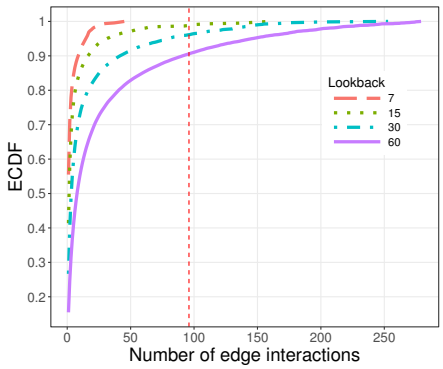


Figure 1. The plot shows the empirical CDF of the number of per edge interactions in Facebook (Events dataset, see Section 5.3).

The main challenge with learning the IC model is that the sample complexity is often overwhelmingly large or simply infeasible. To illustrate this point, Figure 1 shows the cumulative distribution of edge interactions for millions of public events on Facebook over varying periods of time, ranging from one week to two months (see detailed description of the dataset in Section 5). The vertical line marks the minimal number of observations per edge required to infer the likelihood of influence with error 0.1 and confidence 95%. In this data set, more than 90% of the edges do not have enough observations to learn the respective diffusion probabilities accurately, even over a period of two months. Furthermore, in a single week (the timeframe in which inference about an event is often most relevant), *none of the edges in the data set have sufficiently many observations*.

Given that even with the data available on Facebook there are not enough observations to learn the model, one needs to impose additional assumptions. A natural approach is to assume that the diffusion probabilities are a function of network and individuals' characteristics and some underlying

[*]Equal contribution [1]Department of Computer Science, Harvard University [2]Facebook, Menlo Park. Correspondence to: Dimitris Kalimeris <kalimeris@g.harvard.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

global hyperparameter θ. In the case of events for example, it seems reasonable that influence could be estimated as a function of some global unknown multidimensional parameter θ and individuals' characteristics such as location, gender, and age, and topological features like the ratio between the intersection and the union of the individuals' neighborhoods. Using $\mathbf{x}_{u,v}$ to denote the characteristics of u and v, the hyperparametric approach assumes that the probability of u to influence v denoted $p_{u,v}$ is not arbitrary and can be faithfully estimated via some function p that maps θ and $\mathbf{x}_{u,v}$ to $[0,1]$, i.e. $p_{u,v} = p(\theta, \mathbf{x}_{u,v})$. Given a set of characteristics, learning the IC model then reduces to recovering the underlying hyperparameter θ.

Intuitively, learning a hyperparametric model necessitates far fewer samples than a general diffusion model for two main reasons. First, since the diffusion probabilities are correlated, each observation provides information about all edges in the network. Second, it seems reasonable that the sample complexity of learning the hyperparameter should largely depend on the dimension of the hyperparameter rather than the number of edges the network.

A simple example. To solidify our intuition, consider a simple bipartite network $G = (U, V, E)$ where nodes in U attempt to activate nodes from V as depicted in Figure 1.1 and each activation attempt together with its outcome (label) constitutes one sample. Our goal is to find a $\hat{p}_{u,v}$ for every edge $(u, v) \in E$, s.t. with prob. at least $1 - \delta$ for all edges:

$$|p_{u,v} - \hat{p}_{u,v}| \leq \epsilon$$

Hoeffding's inequality and a union bound imply that $\Theta\left(\frac{|E|}{\epsilon^2} \log \frac{|E|}{\delta}\right)$ samples are necessary and sufficient to learn the diffusion probabilities on all the edges in the graph. In comparison, suppose that the diffusion probability of each edge is a function of a hyperparameter $\theta \in [0,1]^d$ and some known features of the edge $\mathbf{x}_{u,v} \in [0,1]^d$ as follows:

$$p_{u,v} = \frac{1}{1 + e^{-\langle \theta, \mathbf{x}_{u,v} \rangle}}$$

Then, learning the diffusion probabilities becomes a logistic regression problem and thus only $\mathcal{O}\left(\frac{d}{\epsilon^2} \log \frac{d}{\delta}\right)$ samples are required, independent of the number of edges. This reduces the sample complexity by a factor of $\frac{|E|}{d}$ which is quite dramatic when the number of edges in the network $|E|$ is large and the dimension of the hyperparameter d is small.

Beyond potential improvements in sample complexity, a hyperparametric model is convenient due to the structure it imposes. A recent line of work on influence maximization in bandit models (Wen et al., 2015; Vaswani et al., 2017), assumes that the diffusion probabilities are a linear function of edge features, i.e. $p_{u,v} = \langle \theta, x_{u,v} \rangle$ and this structure is leveraged in order to develop faster algorithms.

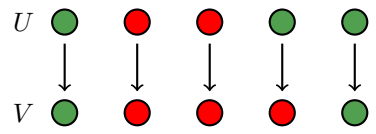


Figure 2. Learning the diffusion probabilities on simple bipartite graphs. Green symbolizes active nodes while red inactive ones.

1.1. A Hyperparametric Approach

Our goal in this paper is to explore a hyperparametric approach for learning the independent cascade diffusion model. Doing so requires addressing three open questions:

- *Does restriction to a low-dimensional hyperparameter substantially decrease the sample complexity?* As discussed above, the motivation for a hyperparametric approach is that *intuitively* its sample complexity should depend on the dimension of the hyperparameter rather than the number of edges. While intuitive, when the indegree of nodes is greater than 1, minimizing empirical risk becomes a non-convex optimization problem and analyzing sample complexity is not trivial;
- *Can a hyperparametric model be learned efficiently?* As learning a hyperparametric model requires solving a non-convex optimization problem, it is not clear it can be learned efficiently, in theory or in practice; [1]
- *Are low-dimensional hyperparametric models predictive?* Assuming that sample complexity heavily depends on its dimension, our approach is relevant only if reasonable estimates of the IC model are achievable with a low-dimensional hypothesis class.

In this paper we address the above questions. We first show that the sample complexity can indeed be dramatically reduced when restricting the hypothesis class to a low-dimensional hyperparameter. Specifically, when comparing with the state-of-the-art bound for learning the independent cascade model (without the hyperparametric assumption) we show that the sample complexity can be reduced by a factor of $|E|/d$, as foreshadowed by the example above.

Despite being a non-concave optimization problem we show that the problem has a great deal of structure. Under mild assumptions about the distribution generating the samples, we show how this structure can be leveraged to efficiently train a model with arbitrarily small generalization error.

Lastly, we show that the hyperparametric approach does work in practice. To do so, we ran experiments on large scale cascades recorded on the Facebook social network. We show that with a hyperparameter of dimension 40 one

[1] We note that even without the hyperparametric assumption PAC learning the IC model is a non-convex optimization problem (Narasimhan et al., 2015).

can estimate the diffusion probabilities with remarkably high accuracy. Naturally, there are data sets that do not contain individuals' characteristics. Nonetheless, most social networking services include useful information about its members that help predict diffusion, and the topology of the network alone often may serve as a good proxy.

2. The Hyperparametric Model

A social network is a finite directed graph $G = (V, E)$, where the set of nodes V represents the individuals in the network and the set of edges E represents their social links.

Independent Cascade (IC) model. The IC model assumes that every node $v \in V$ can either be *active* or *inactive*. All nodes begin as inactive. At time step $t = 0$ a subset of the nodes X called *the seed* becomes active, and activations in the network continue according to the following stochastic process: Each node u that became active at time step $t = \tau$ attempts to influence every one of its neighbors v *only once* at time step $t = \tau + 1$ independently, and succeeds with some probability $p_{u,v}$. A node v that became active during the process will never go back to being inactive. Our work generalizes the standard IC model by assuming that the probabilities $p_{u,v}$ are not arbitrary but correlated and specifically consequences of nodes' features. This is the hyperparametric assumption, as described below.

Hyperparametrization. Every node $u \in V$ is associated with a vector of features containing information about it. Every edge $(u, v) \in E$, is also associated with a feature vector, the concatenation of the feature vectors of its endpoints, denoted by $x_{u,v}$. The diffusion probability of each edge is a function of a global hyperparameter θ and its feature vector. Formally, we assume that there exists a function $p : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$ s.t. $p_{u,v} = p(\theta, x_{u,v})$ for any edge $(u, v) \in E$. In this work we define p as the sigmoid function:

$$p_{u,v} = \sigma(\theta, x_{u,v}) = \frac{1}{1 + e^{-\langle \theta, x_{u,v} \rangle}}.$$

We restrict the hyperparameter θ to lie in a hypothesis class $\mathcal{H} = [-B, B]^d$ for some constant $B > 0$, and w.l.o.g. we assume that the feature vector of every edge lies in $[0, 1]^d$. Additionally, we assume that $p_{u,v}$ is bounded away from 0 and 1 for all edges, i.e. $p_{u,v} \in [\lambda, 1 - \lambda]$, for some $\lambda > 0$.

Further discussion about the hyperparametric model and the choice of the sigmoid function can be found in Appendix A.

Samples. The input to a learning algorithm is a collection of labeled samples. We assume that there is some unknown distribution $\mathcal{D}_0$ over subset of nodes, that activates the initial seed of the cascade, V_0. Subsequently, as we discussed before, we can partition $V \setminus V_0$ into subsets of nodes $V_1, V_2, \ldots, V_{n-1}$ that become activated at steps $\tau = 1, 2, \ldots, n - 1$, respectively [2]. Notice that the cascade can be further decomposed into a sequence of simpler samples as follows: for every $\tau \in \{0, 1, \ldots, n - 1\}$ consider all the nodes $v \notin \cup_{t=0}^{\tau-1} V_t$ that are within distance of 1 from V_τ. For every v that became activated by V_τ (i.e. $v \in V_{\tau+1}$) create the sample $((V_\tau, v), 1)$, and for every v that remained inactive create the sample $((V_\tau, v), 0)$. Throughout this paper we assume that the input to our learning algorithm is of the form $\{(X_i, v_i), y_i\}_{i=1}^m$ where $X_i \subseteq V$ is a subset of active nodes, v_i is a node in distance 1 from X_i and $y_i \in \{0, 1\}$ is its label. In Appendix C we map every seed-generating distribution $\mathcal{D}_0$ to a sample-generating distribution $\mathcal{D}$.

Log-likelihood of a sample. For every node v, the event "v becomes influenced by X, when the hyperparameter has value θ" is a Bernoulli random variable with probability of success $f_v^\theta(X) = 1 - \prod_{u \in X \cap N(v)} (1 - p_{u,v}(\theta))$ where $N(v)$ is the set of *in-neighbors* of node v. Hence, the likelihood of a sample $s = ((X, v), y)$, where $v \notin X$ is $f_v^\theta(X)^y \cdot \big(1 - f_v^\theta(X)\big)^{1-y}$, and the respective log-likelihood of s is:

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

We want to recover a hyperparameter θ that yields accurate estimates. To do so, given a training set $S = \{s_i\}_{i=1}^m$, we seek the most probable hyperparameter generating S by maximizing the cumulative log-likelihood function:

$$\hat{\theta} = \arg\max_{\theta \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(s_i, \theta) \quad (2)$$

Learning a diffusion model. Our goal is to bound the sample complexity, i.e. the number of i.i.d. samples generated by a distribution $\mathcal{D}$ that we need to observe to PAC learn $\mathcal{H}$. That is, guarantee that $\sup_{\theta \in \mathcal{H}} \mathbb{E}_{s \sim \mathcal{D}}[\mathcal{L}(s, \theta)] - \mathbb{E}_{s \sim \mathcal{D}}[\mathcal{L}(s, \hat{\theta})] \leq \epsilon$, with probability at least $1 - \delta$ (see definition of PAC learnability in Appendix B).

Notice that while there are $|E|$ edges in the network, which translates to $|E|$ diffusion probabilities, in the optimization problem (2) there are only d parameters to be learned.

We would like to note that PAC learning guarantees are required to hold for any distribution $\mathcal{D}$ that generates the data. Hence, it is easy to see that the diffusion probabilities or the hyperparameter θ are not learnable without extra assumptions on $\mathcal{D}$. For details refer to Appendix D.

3. Learning a Hyperparametric Model

In this section we prove Theorem 2 which is the main technical result of the paper. The main takeaway is that a hyperparameteric approach makes learning an influence model feasible. Informally, the theorem states that the number of samples required to (ϵ, δ)-PAC learn the model is

[2] The influence process terminates after at most $|V| - 1$ steps.

$\tilde{\mathcal{O}}\left(\Delta^2\left(\frac{\Delta\cdot d+\log\frac{1}{\delta}}{\epsilon^2}\right)\right)$, where Δ is the maximum degree in the network and d is the dimension of the hyperparameter. As we later show in the experiments section, very small *constant* values of d suffice to learn an influence model with almost no error on real data. This is in sharp contrast to the best sample complexity guarantees due to (Narasimhan et al., 2015) for learning the model without the hyperparametric assumption which is $\tilde{\mathcal{O}}\left(\Delta^2\left(\frac{\Delta\cdot|E|+\log\frac{1}{\delta}}{\epsilon^2}\right)\right)$.

Furthermore, imposing assumptions on the distribution $\mathcal{D}$, can reduce the dependence on Δ making the sample complexity (almost) independent of the size of the network.

3.1. Sample Complexity via Radamacher Complexity

The main technical challenge is due to the fact that the MLE objective in (2) is non-concave and we cannot immediately derive sample complexity bounds from convergence guarantees of Stochastic Gradient Descent for example. Instead, to analyze the sample complexity we will argue about the Rademacher complexity of our hypothesis class by using covering numbers. Informally, the Rademacher complexity measures the expressive power of a hypothesis class $\mathcal{H}$ with respect to a probability distribution and the covering number of a set is the number of balls of a certain radius whose union contains the set (see Definitions 2 and 3 in Appendix B). Recall that the sample complexity of a hypothesis class can be derived from its Rademacher complexity.

Theorem 1 ((Shalev-Shwartz & Ben-David, 2014))**.** *Assume that for every sample $s \sim \mathcal{D}$ and every $\theta \in \mathcal{H}$ we have that: $|\mathcal{L}(s,\theta)| \leq C$. Let $S \sim \mathcal{D}^m$ and $\hat{\theta} = \arg\max_{\theta\in\mathcal{H}}\left\{\frac{1}{m}\sum_{i=1}^m \mathcal{L}\left(s_i,\theta\right)\right\}$. Then with probability at least $1-\delta$ over the choice of S we have that:*

$$\mathbb{E}_{s\sim\mathcal{D}}[\mathcal{L}(s,\hat{\theta})] \geq \sup_{\theta\in\mathcal{H}} \mathbb{E}_{s\sim\mathcal{D}}[\mathcal{L}(s,\theta)] - \mathcal{R}(S,\mathcal{H}) - \mathcal{O}\left(C\sqrt{\frac{\log\frac{1}{\delta}}{m}}\right)$$

where $\mathcal{R}(S,\mathcal{H})$ is the Rademacher complexity of the class $\mathcal{H}$ with respect to S.

Hence, our goal reduces to bounding $\mathcal{R}(S,\mathcal{H})$ for a training set S of size m. We do so by discretizing $\mathcal{H}$ by ϵ, and prove that if the discretization is dense enough, then we do not sacrifice a lot by searching for the most likely hyperparameter in the discrete space $\mathcal{H}_\epsilon$ instead of the continuous $\mathcal{H}$.

To this end, we construct an *ϵ-cover* of the hypothesis class $\mathcal{H} = [-B,B]^d$. Proving that the log-likelihood of any fixed sample s, is bounded and Lipschitz[3] in θ with respect to the ℓ_1-norm, where the Lipschitz parameter depends on λ (Lemma 3 in Appendix E), allows us to translate the cover of the space of the hyperparameter, into a cover of the space of the log-likelihood functions, by slightly increasing the number of points we include in it, as stated in Lemma 1. The proof is deferred to Appendix E.

Lemma 1. *Let $S = \{((X_i,v_i),y_i)\}_{i=1}^m$ be a non-empty set of samples and let $\Delta_S = \max_{s\in S}|X\cap N(v)|$ (maximum in-degree of a node that was activated, across all samples). The covering number of the class of all log-likelihood functions for S is $\mathcal{O}\left(\left(\frac{B\rho d}{\lambda^{\Delta_S}\epsilon}\right)^d\right)$, i.e. we can choose a discrete cover $\mathcal{H}_\epsilon \subseteq \mathcal{H}$ of size $\mathcal{O}\left(\left(\frac{B\rho d}{\lambda^{\Delta_S}\epsilon}\right)^d\right)$, such that for all $\theta \in \mathcal{H}$, there exists a $\theta_\epsilon \in \mathcal{H}_\epsilon$ with*

$$\sup_{s\in S}|\mathcal{L}(s,\theta) - \mathcal{L}(s,\theta_\epsilon)| \leq \epsilon.$$

Given the above lemma, we can invoke Massart's lemma (Lemma 5 in Appendix E) on $\mathcal{H}_\epsilon$ which upper bounds the Rademacher complexity of *finite* hypothesis classes. Subsequently, we use Lemma 4 (Appendix E) to upper bound the Rademacher complexity of $\mathcal{H}$ from that of $\mathcal{H}_\epsilon$. We are now ready to prove the main theorem of the section.

Theorem 2 (Sample Complexity of MLE)**.** *Let $G = (V,E)$ be a directed graph and $\mathcal{D}$ be a distribution that generates samples of the form $s = ((X,v),y)$. Let $\Delta = \max_{s\sim\mathcal{D}}|X\cap N(v)|$. Then, for any $\epsilon,\delta \in (0,1)$, if we use Maximum Likelihood Estimation on a training set of size $m \geq m(\epsilon,\delta) = \mathcal{O}\left(\Delta^2\log^2(1/\lambda)\frac{d\log(B\rho d/\lambda^\Delta\epsilon)+\log(1/\delta)}{\epsilon^2}\right)$ samples drawn i.i.d. from $\mathcal{D}$, with probability at least $1-\delta$ (over the draw of the training set) it holds:*

$$\sup_{\theta\in\mathcal{H}} \mathbb{E}_{s\sim\mathcal{D}}[\mathcal{L}(s,\theta)] - \mathbb{E}_{s\sim\mathcal{D}}[\mathcal{L}(s,\hat{\theta})] \leq \epsilon.$$

Proof. Define $\Delta := \max_{s\sim\mathcal{D}}|X\cap N(v)|$, i.e. the maximum active in-degree that any sample generated by $\mathcal{D}$ can have. Then, applying Lemma 1 we can create a discrete ϵ-cover of the space of the log-likelihoods, $\mathcal{H}_\epsilon \subseteq \mathcal{H}$, of size $|\mathcal{H}_\epsilon| = \mathcal{O}\left(\left(\frac{B\rho d}{\lambda^\Delta\epsilon}\right)^d\right)$ for any training set S of any size. Invoking Lemma 4 (Appendix E), we can associate the Rademacher complexity of $\mathcal{H}$ with that of its cover $\mathcal{H}_\epsilon$, for any S and any $\epsilon > 0$, as follows:

$$\mathcal{R}(S,\mathcal{H}) \leq \mathcal{R}(S,\mathcal{H}_\epsilon) + 2\epsilon.$$

Hence, we can focus on bounding the Rademacher complexity of $\mathcal{H}_\epsilon$ instead of that of $\mathcal{H}$. Since $\mathcal{H}_\epsilon$ is finite, the well-known Massart's lemma apply yielding:

$$\mathcal{R}(S,\mathcal{H}) \leq \mathcal{R}(S,\mathcal{H}_\epsilon) + 2\epsilon$$

$$\leq 2\max_{\theta\in\mathcal{H}_\epsilon}\left|\left|\left(\mathcal{L}(s_i,\theta)\right)_{i=1}^m\right|\right|_2 \frac{\sqrt{2\log(|\mathcal{H}_\epsilon|)}}{m} + 2\epsilon$$

$$\leq 2\sqrt{m}\Delta\ln\frac{1}{\lambda}\cdot\frac{\sqrt{2\log(|\mathcal{H}_\epsilon|)}}{m} + 2\epsilon$$

[3] intuitively for a Lipschitz function a small change in the argument cannot lead to a large change in the value of the function, see Definition 4, Appendix B.

$$= 2\Delta \ln \frac{1}{\lambda}\sqrt{\frac{2\log(|\mathcal{H}_\epsilon|)}{m}} + 2\epsilon$$
$$= \mathcal{O}\Big(\Delta \log \frac{1}{\lambda}\sqrt{\frac{d\log(B\rho d/\lambda^\Delta \epsilon)}{m}}\Big) + 2\epsilon$$

where the first inequality holds because of Lemma 4 (discretization), the second because of Massart's lemma (finite hypothesis class), the third because of Lemma 3 (bounded $\mathcal{L}$), and the last one because of Lemma 1 (covering number).

Setting $\epsilon = 1/m$ yields: $\mathcal{R}(S,\mathcal{H}) = \mathcal{O}\Big(\Delta \log(1/\lambda)\sqrt{\frac{d\log(B\rho dm/\lambda^\Delta)}{m}}\Big)$. Now using the generalization bound of Theorem 1, one can see that in order to achieve $\mathbb{E}_{s\sim D}[\mathcal{L}(s,\hat{\theta})] \geq \sup_{\theta\in\mathcal{H}} \mathbb{E}_{s\sim D}[\mathcal{L}(s,\theta)] - \epsilon$, with probability at least $1-\delta$, we need S to be of size $m = \mathcal{O}\Big(\Delta^2 \log^2(1/\lambda)\frac{d\log(B\rho d/\lambda^\Delta\epsilon)+\log(1/\delta)}{\epsilon^2}\Big)$, which concludes the proof. □

Given that B and λ are constants the above bound simplifies to $\tilde{\mathcal{O}}\left(\Delta^2\left(\frac{\Delta\cdot d+\log\frac{1}{\delta}}{\epsilon^2}\right)\right)$, which allows immediate comparison with the bounds derived in (Narasimhan et al., 2015). Additionally, when the degree of every node is constant (which is the case for real social networks like Facebook) or when the distribution $\mathcal{D}$ activates only seeds of constant size, Δ is a constant and the sample complexity becomes $\tilde{\mathcal{O}}\left(\frac{d+\log\frac{1}{\delta}}{\epsilon^2}\right)$, independent of the size of the network.

4. Algorithms

As we mentioned in the previous section, the maximization problem (2) is non-concave and it cannot be solved efficiently. However, the cumulative log-likelihood function we aim to optimize has a great deal of structure we can utilize.

To understand this structure, note that there are only three distinct cases for a sample $s = ((X,v),y)$ in the training set S: (i) node v was not influenced, (ii) node v was influenced and there is only one neighbor of v in X and (iii) node v was influenced and there is more than one neighbor of v in X. The only case that makes the respective log-likelihood non-concave is (iii) since we are unable to identify which of the parents of v actually influenced it and how to update the hyperparameter (equation (1) yields that formally). We refer to such samples as *obfuscated*.

One can partition S into S_o and $S \setminus S_o$ where S_o contains the obfuscated samples. We can then write $\tilde{f}(\theta) := \frac{1}{m}\sum_{s\in S}\mathcal{L}(s,\theta) = \frac{1}{m}\sum_{s\in S\setminus S_o}\mathcal{L}(s,\theta) + \frac{1}{m}\sum_{s\in S_o}\mathcal{L}(s,\theta) =: f(\theta) + \xi(\theta)$. Optimizing $\tilde{f}$ can be perceived as optimizing a concave function f under noise ξ. The magnitude of the noise depends on the probability of seeing an obfuscated sample, which characterizes the difficulty of the optimization problem and can be computed in simple cases (see e.g. Lemma 8 in Appendix F).

There are three distinct approaches that we can follow:

1. Ignore the obfuscated samples and optimize f instead of $\tilde{f}$, using standard methods like Gradient Descent. The fact that the likelihood of each sample is bounded (Lemma 3 in Appendix E) will assure that the recovered solution will approximately optimize $\tilde{f}$ as well.
2. Optimize $\tilde{f}$ directly by applying techniques from (Belloni et al., 2015) for concave optimization under noise.
3. Attempt to optimize $\tilde{f}$ using standard concave optimization techniques (for example Stochastic Gradient Descent (SGD) which is widely used in the training of deep networks, a non-convex optimization problem).

The first two methods provide theoretical guarantees for noise of small magnitude, if the noise is large however, they can lead to large error. See Appendix F for a detailed description of these two approaches. The third heuristic approach works remarkably well in practice, even when the noise is large, as the experiments of Section 5.1 demonstrate. Additionally, in Section 5.2 we include experiments indicating that, even if the noise is small, it is still in our best interest to utilize all the available samples since the shortage of the training set hurts us more than non-concavity.

5. Experiments

We conduct two sets of experiments. First, using synthetic datasets we show that if the hyperparametric assumption holds in a network, we can accurately learn the edge probabilities despite the non-concavity of (2), and significantly outperform methods that do not include information about the node features, for small training sets[4]. We also investigate which properties of the network and the model affect the convergence rate. Secondly, we validate our approach using real Facebook data, by showing that low-dimensional hyperparametric models are predictive in practice.

5.1. Learning the Diffusion Probabilities

Real Graphs: We also use the "ego-facebook", "wiki-Vote", "bitcoin-otc" and "bitcoin-alpha" datasets from (Leskovec & Krevl, 2014), which are publicly available real-world social networks, enabling the reproducibility of our experiments.

Graphs. We synthetically generate the social graph and the hyperparameter that determines the diffusion probabilities.

Synthetic Graphs: We simulate a social network using standard graph models. Since different models yield graphs with different topology, we selected four of the widely used ones: Barabási-Albert, Kronecker, Erdös-Rényi and the con-

[4]Notice that learning the diffusion probabilities allows us to compute other quantities of interest as well, like the probability of a node becoming influenced, the final size of a cascade initiated from a given set, or the influence function.

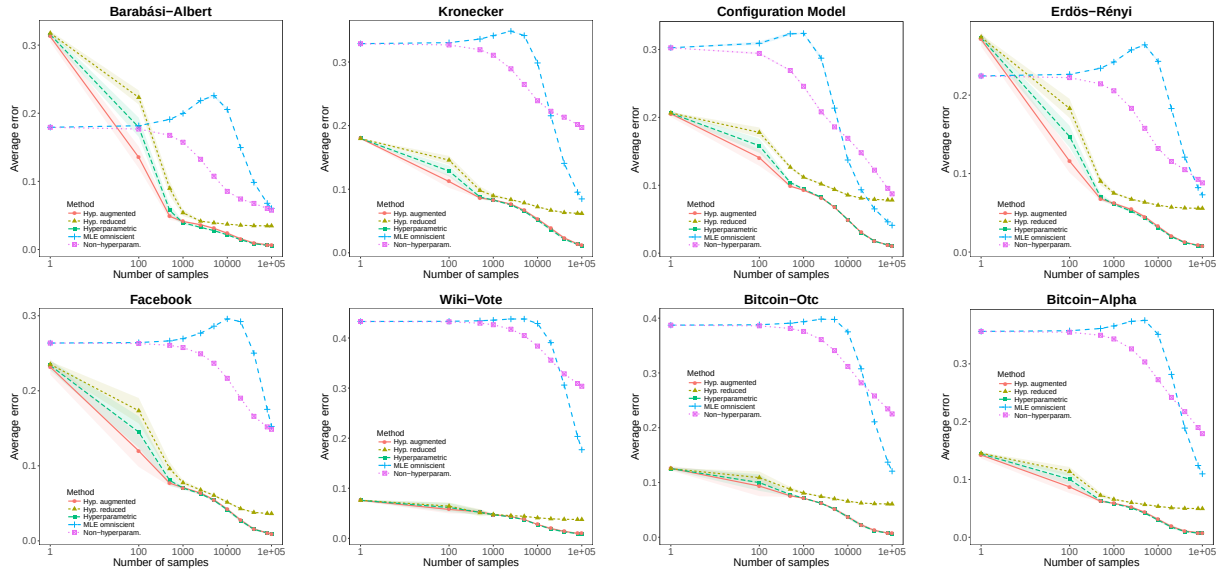


Figure 3. Learning the diffusion probabilities with vs without the hyperparametric assumption.

figuration model. For a more detailed description of these models and the construction process refer to Appendix G.

Experimental setup. We generate 15 random features in $[0, 1]$ for every node (we found consistent results across a large range of features). We define the first 10 of them to be the ones in which the hyperparametric assumption is based, and the rest is redundant information (30 features for each edge, where only 20 of them are important). Additionally, we generate a random hyperparameter in $[-1, 1]^d$ ($d = 20$). We use the sigmoid function over the important features of each edge e and the hyperparameter to compute p_e, as described in Section 2, imposing correlation between the probabilities of different edges by construction.

Subsequently, we generate 100,000 samples, and attempt to solve the optimization problem (2) using SGD, initializing the hyperparameter to 0 and using a learning rate of $1/\sqrt{T}$, where T is the size of the training set. Details on how we create the training set can be found in Appendix G.

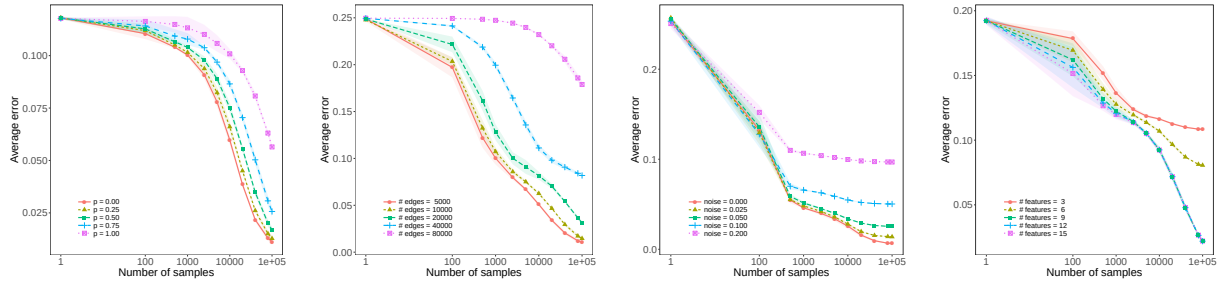
Benchmarks. We tested the hyperparametric model against the following benchmarks:

- *Omniscient MLE:* The true diffusion probability of an edge is approximated by $\hat{p}_e = n_e^+/n_e$, where n_e^+ is the number of activations of edge e, while n_e is the total number of exposures of e (activation attempts). Here, we assume that for every sample $((X, v), y)$ we observe the activation or not of all the edges $e = (u, v)$, where u is an active neighbor of v. This is a strong benchmark since in practice, we can observe whether v became active but not which node activated it.
- *Non-hyperparametric MLE:* We implemented the algorithm of (Narasimhan et al., 2015), that allows one to learn the diffusion probabilities only by observing whether a node v was influenced by the seed X or not.
- *Hyperparametric MLE, reduced information:* We compare ourselves against a hyperparametric model that is unaware of the exact features that are important, and selects only a subset of them. Here we select only 5 out of 10 important features of every node.
- *Hyperparametric MLE, augmented information:* Similarly, we compare ourselves against a hyperparametric model that is unaware of the important features, thus it selects all the available ones (15 features per node).

Results. We repeat each experiment 10 times, and provide the mean and the standard deviation in Figure 3. The y-axis corresponds to $\frac{1}{|E|}\sum_{e\in E}|p_e - \hat{p}_e|$, the average absolute error between the real probability (known by construction) and the empirical one across the network. In all networks, the hyperparametric approach greatly outperforms the non-hyperparametric benchmarks, *even assuming omniscience*.

Note that in the non-hyperparametric benchmarks, since samples do not carry global information, there exist edges that have no exposures given the samples that we have seen so far. In that case, we define $\hat{p}_e = 0$. This explains the initial increase in the error in the omniscient MLE since, if p_e is small and the first exposure of edge e is an activation, the error on e increases from p_e to $1 - p_e$. Once we see enough samples, $\hat{p}_e$ converges to p_e. One can also notice, the effect of omniscience since it leads to faster convergence than actual implementable non-hyperparametric methods.

Regarding the two benchmarks that involve the hyperparametric assumption it is worth noting that reduced information does not allow convergence to 0 error, while augmented



(a) Samples vs Concavity (b) Graph density (c) Noisy model (d) Feature selection

Figure 4. Characteristics of the graph or the model that affect convergence.

information does (see also Figure 4d for a more detailed investigation). Finally, the initial difference in the errors of different networks is related to how good predictor the initialization of SGD is (i.e. $\theta = 0$), as well as how large the average diffusion probability in the network is. The learning effect is evident and universal though, since the error converges to 0 independently of the underlying network.

5.2. Convergence Rate Investigation

In these experiments we use Erdös-Rényi graphs with 1000 nodes and 20000 edges (unless otherwise stated).

Samples vs Concavity. In Section 4 we classified the samples into categories based on whether they yield concave log-likelihood or not. Recall that if we ignore the obfuscated samples, the optimization problem becomes concave. A natural question is whether sacrificing samples for concavity leads to faster convergence. To this end, we generate samples and if a sample is obfuscated, we discard it with probability $p \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$. The results can be found in Figure 4a. It is evident that even though our optimization problem becomes concave and hence theoretically easier to solve, the price due to data shortage is huge.

Approximate models. Here we investigate how properties of the graph or the model affect the convergence rate.

- *Graph Density:* We create an Erdös-Rényi graph with 1000 nodes and varying number of edges, exploring how does graph density affect the convergence of the error. As the density increases, so does the average degree in the network and, as a result, the number of obfuscated samples. Hence, the information obtained becomes "noisier" and convergence is slower.
- *Noisy model:* Until now we assumed that the hyperparametric model is the ground truth. Here we relax this assumption. We generate each edge probability as before, and subsequently add noise uniform in $[-N, N]$, for increasing values of N. Now, the average error does not converge to 0 and increases with N.
- *Features Effect:* In many cases we might not know the exact features that support the hyperparametric model. We explore the effect of this lack of information by including varying number of significant features in our model. Our results show that in terms of convergence more information does not hurt, despite being more costly computationally. However, if we fail to include all the significant features, we do not converge to 0 error, and the error grows with the removal of features.

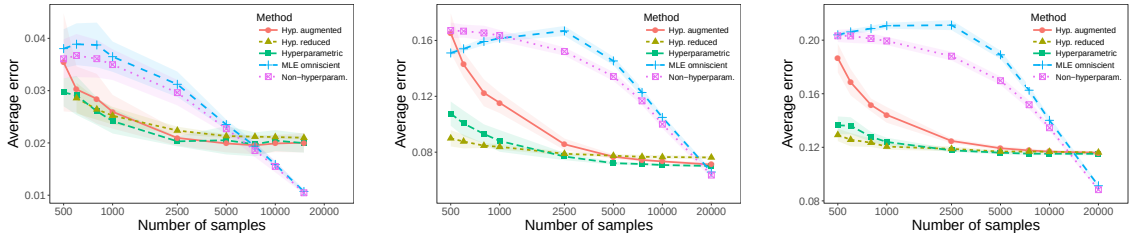
The results can be found in Figure 4b-d. In all the cases the way that we enforced the hyperparametric assumption, created the samples and ran SGD is the same as in Section 5.1.

5.3. Are Low-dimensional Models Predictive?

Importantly, we evaluate the validity of the hyperparametric assumption on real cascade data. To this end, we use the following aggregated and public Facebook data sets containing only de-identified data (i.e. they don't include personally identifying information about individuals in the dataset).

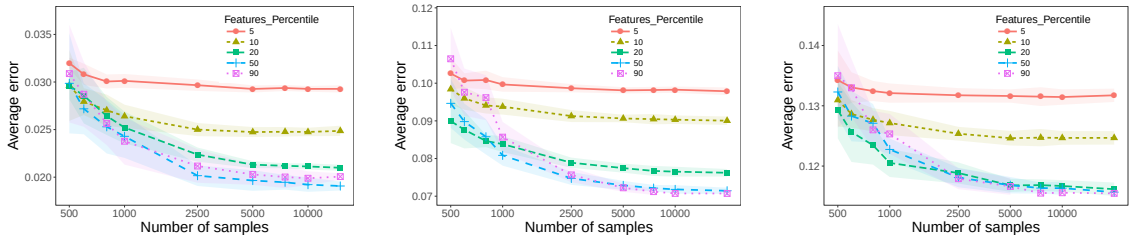
Events. In Facebook a user can invite a set of other users for an event, who can then forward the invite to their friends to join. Moreover, when friends join an event a user may be notified in their Facebook feed and join in turn. The *cascade* in this scenario is an event and an *exposure* is either a direct invite or a feed notification. A user is influenced if she marked herself as "going" to the event. This dataset is a random sample of events that happened over a two-month period in late 2017. We have included only *public events* that are visible to everyone and also excluded users who created the event or joined without being invited. The number of cascades in our dataset is roughly 3 million with 90 million users participating and 130 million exposures.

Video and Photo Reshares. A *cascade* in this dataset is a video or photo content. Whenever a user watches a video (photo) posted from a friend we consider it as an *exposure* and when the user shares that video (photo) after watching we consider it as an *adoption* (we consider only videos that were explicitly seen and not auto-played). We collected a random sample of photo/video data on a random day in January 2018, and included only public photo and video



(a) Events (b) photos (c) Videos

Figure 5. Learning with the hyperparametric assumption in Facebook cascade data sets.



(a) Events (b) photos (c) Videos

Figure 6. Feature sensitivity analysis of hyperparametric model for Facebook data sets.

posts. Our data sets contain roughly 10 million cascades with more than 100 million users and 500 million exposures.

Experimental setup. The features that we include in our model in both cases are user attributes such as Facebook age in days, friend count, number of initiated friendship requests, subscriber count and subscription count, city, country, and language, number of days active in last 7 days, and 28 days. The categorical features were binarized in the model. An important difference with the experiments of Sections 5.1 and 5.2 is that here we don't know the true diffusion probability of every edge by construction. Instead, we estimate it from samples as $\hat{p}_e = \frac{n_e^+}{n_e}$. In order to estimate $\hat{p}_e$ accurately, we need enough samples for edge e. Hence, we restrict our evaluation set (the set of edges where we measure the error) only to edges that have at least 67 interactions, meaning that $|p_e - \hat{p}_e| \leq 0.15$ with probability at least 90%. Each experiment is repeated 50 times and the averages together with the standard deviations are reported in Figures 5 and 6.

Results. Our first set of experiments is to validate the hyperparametric assumption in real data. We observe that using the optimization problem (2), with very few samples, the hyperparametric model achieves significant reduction in average error (up to 60%) over methods that don't utilize node features. The results, reported in Fig. 5, are consistent with our synthetic experiments, where the hyperparametric assumption holds by construction. Note that the non-hyperparametric methods will eventually converge to zero error as they correspond to the ground truth while the hyperparametric model is only a good approximation of it.

We also included reduced and augmented hyperparametric models for comparison as in the synthetic experiments. In the case of the reduced model we used only 20% of the most important features of each edge (measured using Mutual Information). For the augmented version on the other hand, we augment the feature vector of each node with redundant information (increase its dimension by 50% and fill the extra coordinates with random noise) and investigate whether convergence still occurs. As in the experiments of Section 5.1, the reduced model converges to higher average error than the models that use more information, while the augmented model successfully ignores all the redundant features.

We also evaluated the sensitivity of the hyperparametric model when we include all versus few selected features. The picture that we see matches the synthetic experiments (Figure 4d), i.e. the hyperparametric model is supported on several features and if we fail to include all of them our error won't converge to 0. However, an important difference with the synthetic experiments is that here not all the features are equally important, hence by applying feature-selection algorithms we can collect a small subset that performs almost as well as using the entire feature vector (see e.g. the difference in the error between 20% and 90% of the features).

6. Acknowledgements

This research was supported by NSF grant CAREER CCF-1452961, BSF grant 2014389, NSF USICCS proposal 1540428, and a Facebook research award.

References

Abrahao, Bruno D., Chierichetti, Flavio, Kleinberg, Robert, and Panconesi, Alessandro. Trace complexity of network inference. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pp. 491–499, 2013.

Anderson, Ashton, Huttenlocher, Daniel P., Kleinberg, Jon M., Leskovec, Jure, and Tiwari, Mitul. Global diffusion via cascading invitations: Structure, growth, and homophily. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 66–76, 2015.

Belloni, Alexandre, Liang, Tengyuan, Narayanan, Hariharan, and Rakhlin, Alexander. 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.

Bourigault, Simon, Lagnier, Cedric, Lamprier, Sylvain, Denoyer, Ludovic, and Gallinari, Patrick. Learning social network embeddings for predicting information diffusion. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pp. 393–402, 2014.

Du, Nan, Liang, Yingyu, Balcan, Maria, and Song, Le. Influence function learning in information diffusion networks. In *International Conference on Machine Learning*, pp. 2016–2024, 2014.

Friggeri, Adrien, Adamic, Lada, Eckles, Dean, and Cheng, Justin. Rumor cascades. In *International AAAI Conference on Web and Social Media*, 2014.

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

Gomez Rodriguez, Manuel, Leskovec, Jure, and Krause, Andreas. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1019–1028. ACM, 2010.

Goyal, Amit, Bonchi, Francesco, and Lakshmanan, Laks VS. 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.

Kempe, David, Kleinberg, Jon, and Tardos, Éva. 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.

Lazarsfeld, Paul and Merton, Robert K. Friendship as a social process: A substantive and methodological analysis. Morroe Berger, Theodore Abel, and Charles H. Page, editors, Freedom and Control in Modern Society.

Lemonnier, Rémi, Scaman, Kevin, and Vayatis, Nicolas. Tight bounds for influence in diffusion networks and application to bond percolation and epidemiology. In *Advances in Neural Information Processing Systems*, pp. 846–854, 2014.

Leskovec, Jure and Krevl, A. Snap datasets: Stanford large network dataset collection. 2014. URL http://snap.stanford.edu/data.

Leskovec, Jure, Chakrabarti, D, Kleinberg, Jon, and Faloutsos, Christos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *PKDD 2005, volume 3721 of Lecture Notes in Computer Science*, pp. 133–145. Springer, 2005.

Liben-Nowell, David and Kleinberg, Jon M. The link prediction problem for social networks. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003*, pp. 556–559, 2003.

McPherson, Miller, Smith-Lovin, Lynn, and Cook, James M. Birds of a feather: Homophily in social networks. Annual Review of Sociology, 2001.

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

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

Saito, Kazumi, Nakano, Ryohei, and Kimura, Masahiro. Prediction of information diffusion probabilities for independent cascade model. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pp. 67–75. Springer, 2008.

Shalev-Shwartz, Shai and Ben-David, Shai. Understanding machine learning: From theory to algorithms. 2014.

Subbian, Karthik, Prakash, B Aditya, and Adamic, Lada. Detecting large reshare cascades in social networks. In *Proceedings of the 26th International Conference on World Wide Web*, pp. 597–605, 2017.

Vaswani, Sharan, Kveton, Branislav, Wen, Zheng, Ghavamzadeh, Mohammad, Lakshmanan, Laks V. S., and Schmidt, Mark. Diffusion independent semi-bandit influence maximization. In *arXiv preprint*, pp. arXiv:1703.00557, 2017.

Wen, Zheng, Kveton, Branislav, and Ashkan, Azin. Efficient learning in large-scale combinatorial semi-bandits. In *International Conference on Machine Learning*, 2015.

A. The Hyperparametric Model, Discussion

As discussed in Section 2, the essence of the hyperparametric model is that the diffusion probability of every edge is dictated by the features of its endpoints. Node features have been used in several studies and been proved to impact network formation (see e.g. (Lazarsfeld & Merton; McPherson et al., 2001)). Specifically, as a general principle, we tend to be friends with people that are "similar" to us, a phenomenon called *homophily* ("birds of a color flock together"). A recent line of work (Anderson et al., 2015) studies the effect of homophily in diffusion and in cascading behavior in social networks. Of course in real social networks there are connections between very diverse people as well.

The hyperparametric model that we propose in this paper can be viewed as an extension of these observations to the IC model. The intuition is that two nodes with similar features (interests, age, country of residence, etc.) will in principle be more influential to each other than diverse nodes, or similar medical characteristics between two nodes will increase the likelihood of transmitting a disease. This kind of observations are not captured by the traditional IC model, as it is unaware of the individual characteristics of each node and the homophily is present only in the network structure.

A natural question that one can ask is whether the sigmoid function is an appropriate function to use in order to encode the influence probabilities. In principle any function that takes as input two vectors and outputs a value in $[0, 1]$ could work, however the definition of the sigmoid function is very relevant for our purposes, since we can adjust the hyperparameters to capture the changes in the diffusion probabilities as a result of the agreement or the disagreement between the features of different nodes. Specifically, if two nodes have similar value in an important feature, then by choosing the respective coordinates of θ to be small we are increasing the influence probability. Similarly we can decrease the influence probability if the features are very dissimilar. The hyperparametric assumption tells us that there is a θ that is a good compromise over all the nodes in the network. Additionally, assuming that the diffusion probabilities are generated by the sigmoid function, our MLE optimization problem reduces to logistic regression, which is well-understood, in the case where every active node has only one active parent.

B. Definitions

Definition 1 (PAC learnability)**.** A hypothesis class $\mathcal{H}$ is *Probably Approximately Correct* (PAC) learnable with respect to some reward function r, if there exists a function $m_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ and a learning algorithm $\mathcal{A}$ such that for every $\epsilon, \delta \in (0, 1)$ and every data-generating distribution $\mathcal{D}$, if we run $\mathcal{A}$ on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. samples generated by $\mathcal{D}$, it returns a hypothesis $\hat{h}$ such that, with probability at least $1 - \delta$ over the choice of the samples, it holds:

$$\mathbb{E}_{s\sim\mathcal{D}}[r(\hat{h}, s)] \geq \max_{h\in\mathcal{H}} \mathbb{E}_{s\sim\mathcal{D}}[r(h, s)] - \epsilon.$$

Definition 2 (Rademacher Complexity)**.** The Rademacher complexity of a hypothesis class $\mathcal{H}$, with respect to a reward function r and a training set S of size m, drawn from a distribution $\mathcal{D}$ is defined as:

$$\mathcal{R}(\mathcal{H}, S) = \frac{2}{m} \mathbb{E}_{\vec{\sigma}\sim\{-1,1\}^m} \left[\sup_{h\in\mathcal{H}} \sum_{i=1}^{m} \sigma_i r(h, s_i) \right]$$

where $\{-1, 1\}$ symbolizes the uniform distribution with that support.

Definition 3 (Covering Number (Shalev-Shwartz & Ben–David, 2014))**.** Let $A \subseteq \mathbb{R}^d$ be a set of vectors. We say that A is ϵ-covered by a set A_ϵ with respect to some norm p, if for all $\vec{a} \in A$ there exists an $\vec{a}_\epsilon \in A_\epsilon$ such that $||\vec{a} - \vec{a}_\epsilon||_p \leq \epsilon$. The covering number of A is the cardinality of the smallest A_ϵ that ϵ-covers A.

Definition 4 (Lipschitz function)**.** A function $f : A \to B$ is ρ-Lipschitz over A, with respect to some norm α, if for all $x_1, x_2 \in A$ it holds:

$$|f(x_1) - f(x_2)| \leq \rho \, ||x_1 - x_2||_\alpha .$$

C. The Distribution

As discussed in Section 2, a cascade C is a sequence of disjoint subsets of nodes $\{V_0, V_1, \ldots, V_{n-1}\}$ that become active in each time step, where V_0 is the initial seed. Each cascade C is associated with some probability $\mathbb{P}[C]$ that depends on the seed-generating distribution $\mathcal{D}_0$, the structure of the graph and the diffusion probabilities of the edges ($\mathbb{P}[C] = \mathbb{P}[V_0] \cdot \mathbb{P}[V_1|V_0] \cdots \mathbb{P}[V_{n-1}|V_{\tau<n-1}]$).

Given a distribution $\mathcal{D}_0$ that generates the seed we want to define a distribution $\mathcal{D}$ that generates samples of the form $s = ((X, u), y)$ as described in Section 2. We define $\mathcal{D}$ to be the distribution that picks a cascade $C = \{V_0, V_1, \ldots, V_{n-1}\}$ with probability proportional to $\mathbb{P}[C]$ and decomposes it into simpler samples of the form $s = ((X, u), y)$. This decomposition is simple (we already described it in Section 2): for every $\tau \in \{0, 1, \ldots, n-1\}$, consider all the nodes $v \notin \cup_{t=0}^{\tau-1} V_t$ that are within distance of 1 from V_τ. For every v that became activated by V_τ (i.e. $v \in V_{\tau+1}$) create the sample $((V_\tau, v), 1)$, and for every v

that remained inactive create the sample $((V_\tau, v), 0)$. Once the entire list of samples for the cascade C is produced, $\mathcal{D}$ returns one of them uniformly at random.

In other words, every possible sample $s = ((X, u), y)$ is assigned probability $\mathbb{P}[s] = \sum_{C: s \in C} \frac{\mathbb{P}[C]}{\#\text{ samples in } C}$ by $\mathcal{D}$.

Essentially, $\mathcal{D}$ can be thought as a distribution that agrees with $\mathcal{D}_0$ in the creation of the cascade and then picks a single sample out of it that is representative of the entire cascade.

D. Learnability with Respect to the Diffusion Probabilities

The first thing that one might think of is to use the samples provided in order to learn the diffusion probabilities of the edges or the hyperparameter θ itself. However, it is easy to see that this approach will fail. Consider a network with only three nodes, u_1, u_2, v and two edges $(u_1, v), (u_2, v)$ with probabilities 1 and 0 respectively. Consider a sample generating distribution $\mathcal{D}$ that always activates both u_1 and u_2. Then, no matter how many samples drawn from $\mathcal{D}$ we will see, we cannot learn $p_{u_1,v}$ and $p_{u_2,v}$, while we can learn the outcome, i.e. that node v will be influenced. A simple modification shows that the same holds for the learnability of the hyperparameter θ.

Hence, if one wants to achieve convergence to the true diffusion probabilities or the true hyperparameter θ, extra assumptions on the distribution $\mathcal{D}$ as well as the feature vectors of the nodes are required. This is why in this work we are instead interested in a PAC learning guarantee and we use the log-likelihood function, defined in Section 2, as our reward function that allows us to interpret and utilize samples generated by any distribution $\mathcal{D}$.

E. Ommited Lemmas and Proofs, Section 3.1

As we discussed in Section 3.1, the proof of the sample complexity involves covering numbers, but we first need to go through the following two lemmas, that prove the Lipschitz continuity of the local influence function and the log-likelihood. Intuitively, this means that a small change in the argument (hyperparameter) will only impose a small change in the respective log-likelihood function. Hence if we can find a cover for the space of the hyper-parameters, we can convert it into a cover of the space of log-likelihoods with a small increase on its size.

Lemma 2 (Lipschitz Continuity of the Local Influence Function)**.** *The local influence function of any node $v \in V$, $f_v^\theta(X)$, is ρ-Lipschitz for any $X \subseteq V \setminus \{v\}$, i.e. for all $\theta, \theta' \in \mathcal{H} : ||\theta - \theta'||_1 \leq \epsilon \Rightarrow |f_v^\theta(X) - f_v^{\theta'}(X)| \leq \rho\epsilon$, where ρ depends on λ.*

Proof. Fix a node $v \in V$ and an $X \subseteq V \setminus \{v\}$. If we bound the infinite norm of the gradient of $f_v^\theta(X)$ by ρ then this would imply that f is ρ-Lipschitz with respect to the dual norm, i.e. the ℓ_1-norm.

$$\left|\frac{\theta f_v^\theta(X)}{\theta\theta_\ell}\right| = \left|\frac{\theta}{\theta\theta_\ell}\Big(1 - \prod_{u \in X \cap N(v)} (1 - \sigma(\theta, x_{uv}))\Big)\right|$$

$$= \Big|\sum_{u \in X \cap N(v)} \frac{\theta\sigma(\theta, x_{uv})}{\theta\theta_\ell} \prod_{u' \in X \cap N(v): u' \neq u} (1 - \sigma(\theta, x_{u'v}))\Big|$$

$$\leq \sum_{u \in X \cap N(v)} \left|\frac{\theta\sigma(\theta, x_{uv})}{\theta\theta_\ell}\right| \cdot \Big|\prod_{u' \in X \cap N(v): u' \neq u} (1 - \sigma(\theta, x_{u'v}))\Big|$$

$$\leq \sum_{u \in X \cap N(v)} \prod_{u' \in X \cap N(v): u' \neq u} (1 - \lambda)$$

$$= |X \cap N(v)|(1 - \lambda)^{|X \cap N(v)| - 1}$$

where we used the fact that $\left|\frac{\theta\sigma(\theta x_{uv})}{\theta\theta_\ell}\right| \leq 1$, since the sigmoid function σ is 1-Lipschitz as for the ℓ_∞ norm with respect to θ. We also used the fact that the influence probabilities are bounded away from 1.

Now, since the function $h(x) = x(1 - \lambda)^{x-1}$ is maximized for $x = -\frac{1}{\ln(1-\lambda)}$, we get that $f_v^\theta(X)$ is ρ-Lipschitz, for $\rho := \frac{-1}{\ln(1-\lambda)}(1 - \lambda)^{-\left(\frac{1}{\ln(1-\lambda)} + 1\right)}$[5].

□

The Lipschitzness of the local influence function f_v^θ easily implies the Lipschitzness of the log-likelihood of the respective sample.

Lemma 3 (Boundness and Lipschitz continuity of the log–likelihood function)**.** *Fix a hyperparameter $\theta \in \mathbb{R}^d$: $||\theta||_\infty \leq B$. Then, for any valid sample $s = (X, v, y)$ it holds:*

1. $\lambda \leq f_v^\theta(X) \leq 1 - \lambda^{|X \cap N(v)|}$,
2. $|\mathcal{L}(s, \theta)| \leq |X \cap N(v)| \cdot \ln(1/\lambda)$,
3. *$\mathcal{L}(s, \theta)$ is $\frac{\rho}{\lambda^{|X \cap N(v)|}}$-Lipschitz in θ with respect to the ℓ_1 norm.*

Proof. 1. The lower bound is immediate. Since X contains at least one neighbor of v and the minimum influence probability of any edge is λ, node v is influenced

[5] Notice that there is a smooth tradeoff here, if $|X \cap N(v)|$ is very small then a small change in θ will impose a small change in $f_v^\theta(X)$ since there are not many nodes trying to influence v. If $|X \cap N(v)|$ on the other hand is very large then there are so many nodes trying to do so already, that a small change in θ will not have a significant effect on $f_v^\theta(X)$ either.

by the seed X with probability at least λ. For the upper bound, note that in the best case, all the influence probabilities between v and its neighbors would be the maximum possible, i.e. $1-\lambda$. Now, remember that $f_v^\theta(X) = 1 - \prod_{u \in X \cap N(v)}(1 - p_{u,v}(\theta)) \Rightarrow f_v^\theta(X) \leq 1 - \prod_{u \in X \cap N(v)} \lambda \leq 1 - \lambda^{|X \cap N(v)|}$.

2. For a sample $s = ((X, v), y)$ we have:

$$|\mathcal{L}(s,\theta)| = |y \ln\left(f_v^\theta(X)\right) + (1-y)\ln\left(1 - f_v^\theta(X)\right)|$$

$$\leq y|\ln\left(f_v^\theta(X)\right)| + (1-y)|\ln\left(1 - f_v^\theta(X)\right)|$$

$$\leq y|\ln\lambda| + (1-y)|\ln\left(1 - (1 - \lambda^{|X \cap N(v)|})\right)|$$

$$\leq |\ln \lambda^{|X \cap N(v)|}| = |X \cap N(v)| \cdot \ln(1/\lambda)$$

where the second inequality holds since $\lambda \leq f_v^\theta(X) < 1 \Rightarrow \ln\lambda \leq \ln\left(f_v^\theta(X)\right) < 0 \Rightarrow |\ln\lambda| \geq |\ln\left(f_v^\theta(X)\right)|$, and the third inequality since $|X \cap N(v)| \geq 1 \Rightarrow \lambda^{|X \cap N(v)|} \leq \lambda < 1$.

3. Similarly to Lemma 2, we need to bound the ℓ_∞ norm of the gradient of $\mathcal{L}$ with respect to θ. Hence:

$$||\nabla_\theta \mathcal{L}(s,\theta)||_\infty$$

$$= ||\nabla_\theta[y \ln\left(f_v^\theta(X)\right) + (1-y)\ln\left(1 - f_v^\theta(X)\right)]\,||_\infty$$

$$= \left|\left|y \cdot \frac{\nabla_\theta f_v^\theta(X)}{f_v^\theta(X)} - (1-y) \cdot \frac{\nabla_\theta f_v^\theta(X)}{1 - f_v^\theta(X)}\right|\right|_\infty$$

$$\leq \left(\left|y \cdot \frac{1}{f_v^\theta(X)}\right| + \left|(1-y) \cdot \frac{1}{1 - f_v^\theta(X)}\right|\right) \cdot ||\nabla_\theta f_v^\theta(X)||_\infty$$

$$\leq \rho\left(\frac{y}{\lambda} + \frac{1-y}{\lambda^{|X \cap N(v)|}}\right) \leq \frac{\rho}{\lambda^{|X \cap N(v)|}}$$

where the bound on the gradient of f follows from Lemma 2.

□

We are now ready to prove the covering number for the space of the log-likelihood functions. We state Lemma 1 again for completion.

Lemma 1. Let $S = \{((X_i, v_i), y_i)\}_{i=1}^m$ be a non-empty set of samples and let $\Delta_S = \max_{s \in S} |X \cap N(v)|$ (maximum in-degree of a node that was activated across all samples). The covering number of the class of all log-likelihood functions for S is $O\left(\left(\frac{B\rho d}{\lambda^{\Delta_S}\epsilon}\right)^d\right)$, i.e. we can choose a discrete cover $\mathcal{H}_\epsilon \subseteq \mathcal{H}$ of size $O\left(\left(\frac{B\rho d}{\lambda^{\Delta_S}\epsilon}\right)^d\right)$, such that for all $\theta \in \mathcal{H}$, there exists a $\theta_\epsilon \in \mathcal{H}_\epsilon$ with

$$\sup_{s \in S} |\mathcal{L}(s,\theta) - \mathcal{L}(s,\theta_\epsilon)| \leq \epsilon.$$

Proof. Remember that the unknown hyperparameter θ lies in $\mathcal{H} = [-B, B]^d$. Hence, the space of the hyperparameter is a d-dimensional hypercube and it is known that it can be covered by $\left(\frac{Bd}{\epsilon}\right)^d$ ℓ_1-balls of radius ϵ.

Also, in Lemma 3 we proved that for any sample $s = ((X, v), y)$ and for all $\theta, \theta' \in \mathcal{H}$ it holds:

$$|\mathcal{L}(s,\theta) - \mathcal{L}(s,\theta')| \leq \frac{\rho}{\lambda^{|X \cap N(v)|}} ||\theta - \theta'||_1$$

This says that if the hyperparameters are separated by a distance of ϵ in the ℓ_1 space then, for any sample s the likelihoods of it with respect to θ and θ' are within a distance of $\frac{\rho}{\lambda^{|X \cap N(v)|}}\epsilon \leq \frac{\rho}{\lambda^{\max_{s \in S}|X \cap N(v)|}}\epsilon = \frac{\rho}{\lambda^{\Delta_S}}\epsilon$ from each other. Clearly, an ℓ_1 cover of radius ϵ over the parameter space can be translated to a cover of the space of likelihood functions. In particular, if the parameter space is covered by R ℓ_1-balls of radius ϵ and centers $\theta_1, \ldots, \theta_R$, then the likelihood functions $\mathcal{L}^{\theta_1}, \ldots, \mathcal{L}^{\theta_R}$ form a $\frac{\rho}{\lambda^{\Delta_S}}\epsilon$-cover of the space of all the likelihood functions. Thus one can easily see that in order to have an ϵ-cover of the space of the log-likelihoods, we require at most $O\left(\left(\frac{B\rho d}{\lambda^{\Delta_S}\epsilon}\right)^d\right)$ discrete θs. Hence, given the set S, the covering number of the class is $O\left(\left(\frac{B\rho d}{\lambda^{\Delta_S}\epsilon}\right)^d\right)$.

□

The covering number allows us to consider a discrete hypothesis class instead of a continuous one, and hence we can bound its Rademacher complexity, using *Massart's lemma* for finite hypothesis classes. Subsequently, we need to associate the Rademacher complexity of the discretized class with the Rademacher complexity of the continuous one, something that can be done using the following lemma.

Lemma 4 (Discretization Lemma)**.** *Let $\mathcal{H}$ be any hypothesis class and S be a set of m samples drawn from some distribution $\mathcal{D}$, and suppose that $\mathcal{H}_\epsilon$ is an ϵ-cover of S, i.e. for any $h \in \mathcal{H}$ there exists $h_\epsilon \in \mathcal{H}_\epsilon$ such that:*

$$\sup_{s \in S} |r(h,s) - r(h_\epsilon, s)| \leq \epsilon$$

where $r(\cdot,\cdot)$ is some reward function (in our case the log-likelihood). Then it holds:

$$\mathcal{R}(S,\mathcal{H}) \leq \mathcal{R}(S,\mathcal{H}_\epsilon) + 2\epsilon$$

Proof. For any $h \in \mathcal{H}$, let h_ϵ be a hypothesis that covers it. Then, by the definition of the Rademacher complexity it holds:

$$\mathcal{R}(S,\mathcal{H}) = \mathbb{E}_{\vec{\sigma} \sim \{-1,1\}^m}\left[\sup_{h \in \mathcal{H}} \frac{2}{m}\sum_{i=1}^m \sigma_i r(h, s_i)\right]$$

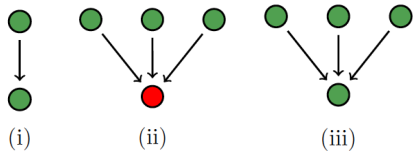


Figure 7. Three different categories of a sample. Nodes in green are active and in red inactive. In case (iii) we know that at least one of the three edges became activated but not which one(s).

$$= \mathbb{E}_{\vec{\sigma}}\Bigg[\sup_{h\in\mathcal{H}}\Big(\frac{2}{m}\sum_{i=1}^{m}\sigma_i r(h_\epsilon, s_i) + \frac{2}{m}\sum_{i=1}^{m}\sigma_i(r(h, s_i) - r(h_\epsilon, s_i))\Big)\Bigg]$$

$$\leq \mathbb{E}_{\vec{\sigma}}\Bigg[\sup_{h\in\mathcal{H}}\frac{2}{m}\sum_{i=1}^{m}\sigma_i r(h_\epsilon, s_i) + \sup_{h\in\mathcal{H}}\frac{2}{m}\sum_{i=1}^{m}\sigma_i(r(h, s_i) - r(h_\epsilon, s_i))\Bigg]$$

$$\leq \mathbb{E}_{\vec{\sigma}}\Bigg[\sup_{h_\epsilon\in\mathcal{H}_\epsilon}\frac{2}{m}\sum_{i=1}^{m}\sigma_i r(h_\epsilon, s_i) + \sup_{h\in\mathcal{H}}\frac{2}{m}\sum_{i=1}^{m}\sigma_i(r(h, s_i) - r(h_\epsilon, s_i))\Bigg]$$

$$= \mathcal{R}(S, \mathcal{H}_\epsilon) + \mathbb{E}_{\vec{\sigma}}\Bigg[\sup_{h\in\mathcal{H}}\frac{2}{m}\sum_{i=1}^{m}\sigma_i(r(h, s_i) - r(h_\epsilon, s_i))\Bigg]$$

$$\leq \mathcal{R}(S, \mathcal{H}_\epsilon) + \mathbb{E}_{\vec{\sigma}}\Bigg[\sup_{h\in\mathcal{H}}\frac{2}{m}\sum_{i=1}^{m}|r(h, s_i) - r(h_\epsilon, s_i)|\Bigg]$$

$$\leq \mathcal{R}(S, \mathcal{H}_\epsilon) + 2\epsilon$$

□

Lemma 5 (Massart's lemma for finite hypothesis classes)**.** *Let $A = \{\vec{\alpha}_1, \vec{\alpha}_2, \ldots, \vec{\alpha}_N\}$ be a finite set of vectors in $\mathbb{R}^m$. Then:*

$$\mathbb{E}_{\vec{\sigma}\sim\{-1,1\}^m}\Big[\max_{\vec{\alpha}\in A}\frac{2}{m}\sum_{t=1}^{m}\sigma_t\alpha_t\Big] \leq 2\cdot\max_{\alpha\in A}||\vec{\alpha}||\frac{\sqrt{2\log N}}{m}$$

F. Solving the Optimization Problem

As we mentioned in Section 4 there are three distinct cases for a sample $s = ((X, v), y)$ in the training set S: (i) node v was not influenced, (ii) node v was influenced and there is only one neighbor of v in X ($|X \cap N(v)| = 1$) and (iii) node v was influenced and there are more than one neighbors of v in X ($|X \cap N(v)| > 1$), as shown in Figure F.

Notice that the likelihood functions corresponding in samples of the kinds (i) and (ii) are concave (by the definition of the log-likelihood, equation (1)). Hence, if there were no obfuscated samples the optimization problem would be concave and thus efficiently solvable via iterative optimization methods such as *Gradient Descent*.

Partitioning S into S_o that contains the obfuscated samples and $S \setminus S_o$ that contains the samples of concave likelihoods, we can express our objective function as $\tilde{f}(\theta) := \frac{1}{m}\sum_{s\in S}\mathcal{L}(s,\theta) = \frac{1}{m}\sum_{s\in S\setminus S_o}\mathcal{L}(s,\theta) + \frac{1}{m}\sum_{s\in S_o}\mathcal{L}(s,\theta) =: f(\theta) + \xi(\theta)$. Optimizing $\tilde{f}$ can be perceived as optimizing a concave function f under noise ξ over a convex set.

The first approach to this problem is to ignore the obfuscated samples (i.e. the noise) and optimize f instead of $\tilde{f}$ using Gradient Descent. The success of this approach lies in the fact that the log-likelihood of each sample is bounded hence, if we have a small number of obfuscated samples the maximizer of f will approximately maximize $\tilde{f}$ as well.

Lemma 6. *Let m_o denote the number of obfuscated samples in a training set S of m i.i.d. samples drawn from $\mathcal{D}$. If $\frac{m_o}{m} \leq \frac{\epsilon}{\Delta_S \ln(1/\lambda)}$, and we use Gradient Descent on $f(\theta) = \frac{1}{m}\sum_{s\in S\setminus S_o}\mathcal{L}(s,\theta)$ for $T \geq \left(\frac{Bd\rho}{\lambda^{\Delta_S}\epsilon}\right)^2$ iterations with a learning rate of $\eta = \sqrt{\frac{B^2\lambda^{2\Delta_S}}{\rho^2 T}}$, we can recover $\hat{\theta} \in [-B, B]^d$ such that:*

$$\tilde{f}(\hat{\theta}) \geq \max_{\theta\in\mathcal{H}}\tilde{f}(\theta) - 2\epsilon.$$

Proof. To simplify the notation in this proof let $\theta^* = \arg\max_\theta f(\theta)$ and $\tilde{\theta} = \arg\max_\theta \tilde{f}(\theta)$.

The first thing to notice is that, as we argued before, f is a concave function over a convex set, hence it can be approximately optimized using GD (note that GD is used for minimization problems but since f is concave, $-f$ is a convex function over a convex set and the minimum of $-f$ is the same as the maximum of f).

Also, since the function f is $\frac{\rho}{\lambda^{\Delta_S}}$-Lipschitz with respect to the ℓ_1 norm, it is $\frac{\rho\sqrt{d}}{\lambda^{\Delta_S}}$-Lipschitz with respect to the ℓ_2 norm. Additionally, it holds: $||\theta||_2 \leq B\sqrt{d}$.

Known results on the convergence of GD (see e.g. (Shalev-Shwartz & Ben-David, 2014)), imply that running GD for $T \geq \left(\frac{Bd\rho}{\lambda^{\Delta_S}\epsilon}\right)^2$ iterations using a learning rate of $\eta = \sqrt{\frac{B^2\lambda^{2\Delta_S}}{\rho^2 T}}$ will return a $\hat{\theta} \in \mathcal{H}$ such that:

$$f(\hat{\theta}) \geq f(\theta^*) - \epsilon.$$

Now, we will use the fact that the noise ξ is small to show that the maximizer of the concave function f, is an approximate maximizer for the approximate concave function $\tilde{f}$:

$$\begin{aligned}\tilde{f}(\tilde{\theta}) &= f(\tilde{\theta}) + \xi(\tilde{\theta}) \leq f(\theta^*) + \xi(\tilde{\theta}) \\ &\leq f(\hat{\theta}) + \epsilon + \xi(\tilde{\theta}) \\ &\leq f(\hat{\theta}) + \xi(\hat{\theta}) - \xi(\hat{\theta}) + \epsilon \\ &= \tilde{f}(\hat{\theta}) - \xi(\hat{\theta}) + \epsilon \\ \Rightarrow \tilde{f}(\hat{\theta}) &\geq \tilde{f}(\tilde{\theta}) - \frac{m_o}{m}\Delta_S \ln\frac{1}{\lambda} - \epsilon \\ &\leq \tilde{f}(\tilde{\theta}) - 2\epsilon\end{aligned}$$

where the first inequality holds since θ^* is the maximizer of f, the second because of the GD guarantee and the third because the log-likelihood of any sample is always negative, hence $\xi(\theta) < 0$, for every $\theta \in \mathcal{H}$. From Lemma 3 we know that $\forall s \in S, \forall \theta \in \mathcal{H} : |\mathcal{L}(s,\theta)| \leq \Delta_S \ln\frac{1}{\lambda}$. Hence, using triangle inequality we can get:

$$\left|\xi(\hat{\theta})\right| = \left|\frac{1}{m}\sum_{s\in S_o}\mathcal{L}(s,\hat{\theta})\right| \leq \frac{m_o}{m}\Delta_S \ln\frac{1}{\lambda}.$$

Finally, the last inequality holds because of the assumption that: $\frac{m_o}{m} \leq \frac{\epsilon}{\Delta_S \ln(1/\lambda)}$. □

Note that in real social networks Δ_S is constant so the running time of GD is $\mathcal{O}\left(\frac{d^2}{\epsilon^2}\right)$. However, even in cases where we have a few nodes with super-constant degree, we can consider the respective samples as noise (hence add them to S_o) and still run GD in polynomial time at the price of slightly increased error, due to the increase in the noise.

Corollary 1 (Efficient Learnability)**.** *Let $G = (V, E)$ be a directed graph and $\mathcal{D}$ be a distribution that generates samples of the form $s = ((X, v), y)$. Let $\Delta = \max_{s\sim\mathcal{D}} |X \cap N(v)|$. Then, for any $\epsilon, \delta \in (0, 1)$, if we use Maximum Likelihood Estimation on a training set of size $m \geq m(\epsilon,\delta) = \mathcal{O}\Big(\Delta^2 \log^2(1/\lambda)\frac{d\log(B\rho d/\lambda^\Delta\epsilon)+\log(1/\delta)}{\epsilon^2}\Big)$ samples drawn i.i.d. from $\mathcal{D}$, and $\frac{m_o}{m} \leq \frac{\epsilon}{\Delta_S \ln(1/\lambda)}$, then with probability at least $1-\delta$ (over the draw of the training set) it holds:*

$$\sup_{\theta\in\mathcal{H}} \mathbb{E}_{s\sim\mathcal{D}}[\mathcal{L}(s,\theta)] - \mathbb{E}_{s\sim\mathcal{D}}[\mathcal{L}(s,\hat{\theta})] \leq 3\epsilon.$$

Moreover, the MLE runs in time polynomial in d and ϵ.

Proof. Follows from the proof of Theorem 1 and the fact that $\hat{\theta}$ approximately optimizes the cummulative log-likelihood over S up to an additive term of 2ϵ, according to Lemma 7. □

We now focus on the second approach: optimize $\tilde{f}$ directly.

Corollary 2 (Using (Belloni et al., 2015))**.** *Let m_o denote the number of obfuscated samples in a training set S of m i.i.d. samples. Then if $\frac{m_o}{m} \leq \frac{\epsilon}{d\Delta_S \ln(1/\lambda)}$, there is a randomized algorithm that can recover $\hat{\theta} \in [-B, B]^d$ such that:*

$$\mathbb{E}\left[\tilde{f}(\hat{\theta})\right] \geq \max_{\theta\in\mathcal{H}} \tilde{f}(\theta) - 2\epsilon.$$

Proof. From Lemma 3 we know that $\forall s \in S, \forall \theta \in \mathcal{H} : |\mathcal{L}(s,\theta)| \leq \Delta_S \ln\frac{1}{\lambda}$. Hence, using triangle inequality we can get:

$$|\xi(\theta)| = \left|\frac{1}{m}\sum_{s\in S_o}\mathcal{L}(s,\theta)\right| \leq \frac{m_o}{m}\Delta_S \ln\frac{1}{\lambda}.$$

So, for $\frac{m_o}{m} \leq \frac{\epsilon}{d\Delta_S \ln(1/\lambda)}$ it holds $|\xi(\theta)| \leq \frac{\epsilon}{d}$, for all $\theta \in \mathcal{H}$. Also note that the convex set $\mathcal{H} = [-B, B]^d$ is well-rounded, according to the definition of (Belloni et al., 2015), because it is contained between the d-dimensional ball of radius B and the one of radius $B\sqrt{d}$, and that there is a trivial membership oracle to $\mathcal{H}$ (just check whether all coordinates of a vector are in $[-B, B]$).

Finally, note that since $\mathcal{L}$ is $\frac{\rho}{\lambda^\Delta}$-Lipschitz with respect to the ℓ_1 norm, f is also $\frac{\rho}{\lambda^\Delta}$-Lipschitz with respect to the ℓ_1 norm and, as a consequence, $\frac{d\rho}{\lambda^\Delta}$−Lipschitz with respect to the ℓ_∞ norm. Hence, all the requirements of the algorithm of (Belloni et al., 2015) are satisfied, and we can apply Simulated Annealing to recover a vector of hyperparameters $\hat{\theta} \in \mathcal{H}$ such that on expectation it holds:

$$\tilde{f}(\hat{\theta}) \geq \max_{\theta\in\mathcal{H}} \tilde{f}(\theta) - 2\epsilon$$

which completes the proof. □

Since for large enough training set S, the value $\frac{m_o}{m}$ will converge to the real probability p_o of seeing an obfuscated sample, the results above essentially tell us that if p_o is small enough we can still optimize the function and recover the hyperparameter despite the non-concavity of the objective function. Hence p_o quantifies the "difficulty" of the optimization problem. It depends on the distribution that generates the samples and it can be bounded in simple cases.

The following lemma provides an upper bound on that probability for the case where each node is chosen to participate in X independently with probability p_X. More involved analysis is possible for different sample-generating distributions.

Lemma 7. *Let $G = (V, E)$ be a graph and $s = ((X, v), y)$ be a sample where each node of V is chosen to participate in X independently with probability p_X. The probability p_o that s is obfuscated, is upper bounded by $1 - \big(1 - p_X \cdot$*

$(1-\lambda))^{\Delta} - p_X(1-p_X)^{\Delta-1} \cdot \lambda$, *where* Δ *is the maximum degree in the graph.*

Proof. We want to bound the probability that we will get an obfuscated sample, i.e. a sample for which $y = 1$ and $|X \cap N(v)| > 1$ assuming that $v \notin X$. It holds:

$$\mathbb{P}[y=1|v \notin X] = \mathbb{P}[(y=1) \cap (|N(v) \cap X| > 1)|v \notin X] + \mathbb{P}[(y=1) \cap (|N(v) \cap X| \leq 1)|v \notin X]$$
$$\Downarrow$$
$$\mathbb{P}[(y=1) \cap (|N(v) \cap X| > 1)|v \notin X] = \mathbb{P}[y=1|v \notin X] - \mathbb{P}[(y=1) \cap (|N(v) \cap X| \leq 1)|v \notin X]$$

So to compute the probability of getting an obfuscated sample, we need to compute the probability of a node becoming active given that it does not belong in X, and the probability of becoming active while having at most one parents in X (given that it does not belong in X).

Let's fist upper bound the probability of node v becoming active. Remember that each node is selected to participate in X with probability p_X, and that for the influence probability of each edge $e \in E$ holds $p_e \in [\lambda, 1-\lambda]$. Hence:

$$\mathbb{P}[y=1|v \notin X] = 1 - \prod_{u \in N(v)} \big(1 - \mathbb{P}[u \text{ activates } v]\big)$$
$$= 1 - \prod_{u \in N(v)} \big(1 - p_X \cdot p_{u,v}\big)$$
$$\leq 1 - \big(1 - p_X \cdot (1-\lambda)\big)^{\Delta}$$

It remains to lower bound the probability that v becomes active while having only one active parent. It is:

$$\mathbb{P}[(y=1) \cap (|N(v) \cap X| \leq 1)|v \notin X]$$
$$= \mathbb{P}[(y=1) \cap (|N(v) \cap X| = 0)|v \notin X] + \mathbb{P}[(y=1) \cap (|N(v) \cap X| = 1)|v \notin X]$$
$$= 0 + \mathbb{P}[(y=1) \cap (|N(v) \cap X| = 1)|v \notin X]$$
$$= \sum_{u \in N(v)} p_X(1-p_X)^{|N(v)|-1} \cdot p_{u,v}$$
$$= p_X(1-p_X)^{|N(v)|-1} \cdot \sum_{u \in N(v)} p_{u,v}$$
$$\geq p_X(1-p_X)^{\Delta-1} \cdot \lambda$$

Putting everything together we get the desired upper bound:

$$\mathbb{P}[(y=1) \cap (|N(v) \cap X| > 1)|v \notin X] \leq 1 - \big(1 - p_X \cdot (1-\lambda)\big)^{\Delta} - p_X(1-p_X)^{\Delta-1} \cdot \lambda$$

□

G. Omitted Details from the Experiments

Synthetic Graphs: As we discussed in Section 5.1 different graph models yield graphs with different topological properties. The ones we selected for our experiments are the following:

- *Barabási-Albert:* The degree distribution of this model is a power law and hence captures interesting properties of the real-world social networks. We took 10 initial vertices and added 10 edges at each step, using the preferential attachment model, until we reached 1000 vertices.
- *Kronecker graphs:* This model for social networks was introduced in (Leskovec et al., 2005). The adjacency matrix of a Kronecker graph is generated by repeated applications of the Kronecker product to an initial seed matrix. In this case we started from a star graph with 4 vertices and computed the Kronecker product till we reached 1000 vertices.
- *Configuration model:* The configuration model allows us to construct a graph with a given degree distribution. We chose 1000 vertices and a power-law degree distribution with parameter $\alpha = 2$.
- *Erdös-Rényi:* We used the celebrated $G(n, m)$ model to create a graph with 1000 vertices and 20000 edges. $G(n, m)$ does not capture some of the properties of real social networks, however it is a very impactful model with variety of applications in several areas of science.

Training Set. We randomly activate an initial seed X of size 10% of the size of the network. X is chosen large to ensure that there exist nodes with multiple active parents and study whether convergence occurs even when (2) is indeed non-concave. We choose one node v reachable from the seed X uniformly at random. If v becomes influenced by X its label y is set to 1, and to 0 otherwise. The seed X, together with v and the label y form one sample $s = ((X, v), y)$ as described in Section 3. We generate 100,000 such samples and attempt to solve the optimization problem (2) using SGD, initializing the hyperparameters to 0 and using a learning rate of $1/\sqrt{T}$, where T is the number of iterations.