Influence Function Learning in Information Diffusion Networks

Nan Du, Yingyu Liang {DUNAN,YLIANG39}@GATECH.EDU
Maria-Florina Balcan NINAMF@CC.GATECH.EDU
Le Song LSONG@CC.GATECH.EDU
College of Computing, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, 30332 USA

Abstract

Can we learn the influence of a set of people in a social network from cascades of information diffusion? This question is often addressed by a two-stage approach: first learn a diffusion model, and then calculate the influence based on the learned model. Thus, the success of this approach relies heavily on the correctness of the diffusion model which is hard to verify for real world data. In this paper, we exploit the insight that the influence functions in many diffusion models are coverage functions, and propose a novel parameterization of such functions using a convex combination of random basis functions. Moreover, we propose an efficient maximum likelihood based algorithm to learn such functions directly from cascade data, and hence bypass the need to specify a particular diffusion model in advance. We provide both theoretical and empirical analysis for our approach, showing that the proposed approach can provably learn the influence function with low sample complexity, be robust to the unknown diffusion models, and significantly outperform existing approaches in both synthetic and real world data.

1. Introduction

Social networks are important in information diffusion, which has motivated the influence maximization problem: find a set of nodes whose initial adoptions of an idea can trigger the largest number of follow-ups. This problem has been studied extensively in literature from both modeling and algorithmic point of view (Kempe et al., 2003; Chen et al., 2010; Borgs et al., 2012; Rodriguez & Schölkopf, 2012; Du et al., 2013b). Essential to the influence maximization problem is the influence function of a set of nodes, which is an estimate of the expected number of triggered follow-ups from these nodes.

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

In practice, the influence function is not given to us, and we only observe the information diffusion traces, or cascades, originating from these nodes. In order to model the cascade data, many information diffusion models have been proposed in the literature, such as the discrete-time independent cascade model and linear threshold model (Kempe et al., 2003), and more recently the continuous-time independent cascade model (Gomez Rodriguez et al., 2011; Du et al., 2013b). To estimate the influence, we can employ a two-stage method: a particular diffusion model is first learned from cascade data, and then the influence function is evaluated or approximated from such learned model.

However, there still remain many challenges in these traditional two-stage approaches. First, real world information diffusion is complicated, and it is not easy to determine the most suitable diffusion model in practice. A chosen diffusion model may be misspecified compared to real world data and lead to large model bias. Second, the diffusion network structure can be also hidden from us, so we need to learn not only the parameters in the diffusion model, but also the diffusion network structure. This often leads to under-determined high dimensional estimation problem where specialized methods need to be designed (Du et al., 2012; 2013a). Third, calculating the influence based on learned diffusion models often leads to difficult graphical model inference problem where extra approximation algorithms need to be carefully designed (Du et al., 2013b).

If the sole purpose is to estimate the influence, can we avoid the challenging diffusion model learning and influence computation problem? In this paper, we provide a positive answer to the question and propose an approach which estimates the influence function directly from cascade data. Our approach will exploit the observation that the influence functions in many diffusion models are coverage functions. Instead of learning a particular diffusion model, we will aim to learn a coverage function instead, which will then naturally subsume many diffusion models as special cases. Furthermore, in the information diffusion context, we show that the coverage function can be represented as a sum of simpler functions, each of which is an expectation over random binary functions. Based on these structures of the problem, we propose a maximum-likelihood based approach to learn the influence function directly from cascade data. More precisely,

Direct and robust approach. Our algorithm does not rely on the assumption of a particular diffusion model, and can be more robust to model misspecification than two-stage approaches. Furthermore, directly learning the coverage function also allows us to avoid the difficulty involved in diffusion model estimation and influence computation.

Novel Parameterization. We propose a parametrization of the coverage function using a convex combination of random basis function. Similar parameterization has been used in classification and kernel methods setting (Rahimi & Recht, 2008), but its usage in the information diffusion and coverage function estimation context is novel.

Approximation guarantee. We show that our parameterization using K random basis functions generates a rich enough family of functions which can approximate the true influence function within an error of $O(\frac{1}{\sqrt{K}})$. This allows us to work with a small number of parameters without creating too much bias at the same time.

Efficient algorithm. We propose a maximum likelihood based convex formulation to estimate the parameters, which allows us to leverage existing convex optimization techniques (Kivinen & Warmuth, 1997) to solve the problem efficiently. The time required to evaluate each gradient is $O(dmK)$, linear in the number of nodes d, the number of cascades m, and the number of basis functions K.

Sample complexity. We prove that to learn the influence function to an ϵ error, we only need $O(\frac{d^3}{\epsilon^3})$ cascades where d is the number of nodes in the diffusion networks. This is no obvious since the number possible source configurations can be exponential in the number of nodes in the network. Our approach is able to make use of the structure of the coverage function and be able to generalize only after seeing a polynomial number of cascades.

Superior performance. We evaluate our algorithms using large-scale datasets, and show that it achieves significantly better performance in both the synthetic cases where there is known model misspecification, and in real world data where the true model is completely unknown in advance.

2. Diffusion Models and Influence Function

Several commonly used models exist for information diffusion over networks. Interestingly, although these models are very different in nature, the derived influence functions belong to the same type of combinatorial functions — coverage functions. Such commonality allows us later to approach the problem of learning influence functions directly without assuming a particular diffusion model.

More specifically, a diffusion model is often associated with a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and a cascade from a model is just a set of influenced nodes according to the model given a set of source nodes $\mathcal{S} \subseteq \mathcal{V}$. In general, we have the following typical types of diffusion models :

Discrete-time independent cascade model (Kempe et al., 2003). Each edge is associated with a weight in $[0, 1]$. When a cascade is being generated from the source nodes $\mathcal{S}$, independently for each edge according to the edge weight, a binary random variable is sampled, indicating whether the edge is included in a "live edge graph" or not. The influenced nodes are those reachable from at least one of the source nodes in the resulting "live edge graph".

Discrete-time linear threshold model (Kempe et al., 2003). Each edge is again associated with a weight in $[0, 1]$, but the sum of the incoming edge weights for each node is smaller or equal to 1. When a cascade is being generated from the source nodes $\mathcal{S}$, each node first independently sample one of its incoming edges with probability proportional to the edge weight. The chosen edges are then used to form the "live edge graph". The influenced nodes are those reachable from at least one of the source nodes.

Continuous-time independent cascade model (Du et al., 2013b). Being different from the discrete-time models, this model associates each edge (j, i) with a transmission function, $f_{ji}(\tau_{ji})$, a density over time. The source nodes are assumed to be initially influenced at time zero. Then a diffusion time is sampled independently for each edge according to the transmission function and is viewed as the length of the edge. The influenced nodes are those within shortest distance T from at least one of the source nodes.

Being common to these diffusion models, the influence function, $\sigma(\mathcal{S}) : 2^{\mathcal{V}} \mapsto \mathbb{R}_+$, of a set of nodes $\mathcal{S}$, is defined as the expected number of influenced nodes with respect to the generative process of each model. This influence function is a combinatorial function which maps a subset $\mathcal{S}$ of $\mathcal{V}$ to a nonnegative number.

Although these diffusion models are very different in nature, their corresponding influence functions belong to the same type of functions — *coverage functions*, and share very interesting combinatorial structures (Kempe et al., 2003; Rodriguez & Schölkopf, 2012). This means that the influence function can be written as

$$\sigma(\mathcal{S}) = \sum\nolimits_{u \in \bigcup_{s \in \mathcal{S}} \mathcal{A}_s} a_u, \qquad (1)$$

with three sets of objects :

(*i*) a ground set $\mathcal{U}$ which may be different from the set $\mathcal{V}$ of nodes in the diffusion network,
(*ii*) a set of nonnegative weights $\{a_u\}_{u \in \mathcal{U}}$, each associated with an item in the ground set $\mathcal{U}$,
(*iii*) and a collection of subsets $\{\mathcal{A}_s : \mathcal{A}_s \subseteq \mathcal{U}\}_{s \in \mathcal{V}}$, one for each source node in diffusion network.

Essentially, each source node $s \in \mathcal{S}$ covers a set $\mathcal{A}_s$ of items from $\mathcal{U}$, and the function value $\sigma(\mathcal{S})$ is the weighted sum over the union of items covered by all nodes in $\mathcal{S}$.

The combinatorial structures of coverage functions allow them to be potentially learned directly from cascades. However, the problem of learning coverage functions is very challenging for several reasons. First, there are an exponential number of different $\mathcal{S}$ from the power set of $\mathcal{V}$, while one typically only observes a small number of cascades polynomial in the number of nodes, $d = |\mathcal{V}|$, in the network. Second, both the ground set $\mathcal{U}$, the weights $\{a_u\}$ and the subsets $\{\mathcal{A}_s\}$ are unknown, and one has to estimate a very large set of parameters if one wants to use the definition in (1) directly.

In fact, learning such combinatorial functions in general settings has attracted many recent research efforts (Balcan & Harvey, 2011; Badanidiyuru et al., 2012; Feldman & Kothari, 2013; Feldman & Vondrak, 2013), many of which show that coverage functions can be learned from just polynomial number of samples. However, existing algorithms are mostly of theoretical interest and impractical for real world problem yet. To tackle this challenge, we will exploit additional structure of the coverage function in the information diffusion context which allows us to derive compact parameterization of the function, and design a simple and efficient algorithm with provable guarantees.

3. Structure of the Influence Function

Besides being coverage functions, the influence functions, $\sigma(\mathcal{S})$, in the diffusion models discussed in Section 2 share additional structures. In all models, a random graph $\mathscr{G}$ is first sampled from the distribution induced by a particular diffusion model; and then a function is defined for computing node reachability in the sampled graph; finally the influence is the expectation of this function with respect to the distribution of the random graphs.

3.1. Random reachability function

We represent each sampled random graph $\mathscr{G}$ as a binary reachability matrix $\boldsymbol{R} \in \{0,1\}^{d\times d}$ with (s,j)-th entry

$$\boldsymbol{R}_{sj} = \begin{cases} 1, & j \text{ is reachable from source } s, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Essentially, the s-th row of $\boldsymbol{R}$, denoted as $\boldsymbol{R}_{s:}$, records the information that if s is the source, which node is reachable given sampled graph $\mathscr{G}$. Furthermore, the j-th column of $\boldsymbol{R}$, denoted as $\boldsymbol{R}_{:j}$, records the information that whether j is reachable from each of the other nodes. Then given a set $\mathcal{S}$ of sources, we can calculate whether a node j will be influenced or not in graph $\mathscr{G}$ through a simple nonlinear function ϕ defined below.

First, we represent the set $\mathcal{S}$ as an indicator vector $\chi_{\mathcal{S}} \in \{0,1\}^d$, with its i-th entry

$$\chi_{\mathcal{S}}(s) := \begin{cases} 1, \ s \in \mathcal{S}, \\ 0, \ \text{otherwise.} \end{cases} \tag{3}$$

Then the inner product $\chi_{\mathcal{S}}^\top \boldsymbol{R}_{:j} \in \mathbb{Z}_+$ will give us an indication whether a target node j is reachable from any of the sources in $\mathcal{S}$. More specifically, $\chi_{\mathcal{S}}^\top \boldsymbol{R}_{:j} \geq 1$ if the target node j is reachable, and 0 otherwise. Finally, using a concave function $\phi(u) = \min\{u,1\} : \mathbb{Z}_+ \mapsto \{0,1\}$, we can transform $\chi_{\mathcal{S}}^\top \boldsymbol{R}_{:j}$ into a binary function of $\chi_{\mathcal{S}}$

$$\phi\left(\chi_{\mathcal{S}}^\top \boldsymbol{R}_{:j}\right) \quad : \quad 2^{\mathcal{V}} \mapsto \{0,1\}. \tag{4}$$

We note that $\phi\left(\chi_{\mathcal{S}}^\top \boldsymbol{R}_{:j}\right)$ itself is a coverage function where (*i*) the ground set $\mathcal{U}$ contains a single item u_j, (*ii*) the weight on u_j is 1, (*iii*) and the collection of subset is either $\mathcal{A}_s = \{u_j\}$ if $\boldsymbol{R}_{sj} = 1$ and otherwise $\mathcal{A}_s = \emptyset$ if $\boldsymbol{R}_{sj} = 0$.

Then the influence of $\mathcal{S}$ in graph $\mathscr{G}$ is the number of target nodes reachable from the source set $\mathcal{S}$

$$\#(\mathcal{S}|\boldsymbol{R}) := \sum_{j=1}^{d} \phi\left(\chi_{\mathcal{S}}^\top \boldsymbol{R}_{:j}\right). \tag{5}$$

$\#(\mathcal{S}|\boldsymbol{R})$ is also a coverage function where (*i*) the ground set $\mathcal{U}$ contains d items $u_1,\ldots,u_d$, (*ii*) the weight on each u_j is 1, (*iii*) and $\mathcal{A}_s = \{u_j \,|\, \boldsymbol{R}_{sj} = 1\}$. Since the graph $\mathscr{G}$ and the associated $\boldsymbol{R}$ are random quantities, the Φ function is a random function.

3.2. Expectation of random functions

Each diffusion model will induce a distribution over random graph $\mathscr{G}$ and hence a distribution $p_{\boldsymbol{R}}$ over the random binary matrix $\boldsymbol{R}$. Then the overall influence of a source set $\mathcal{S}$ in a diffusion model is the expected value of $\#(\mathcal{S}|\boldsymbol{R})$, *i.e.*,

$$\sigma(\mathcal{S}) := \mathbb{E}_{\boldsymbol{R}\sim p_{\boldsymbol{R}}}\left[\#(\mathcal{S}|\boldsymbol{R})\right], \tag{6}$$

which is also a coverage function, since non-negative combinations of coverage functions are still coverage functions (See Appendix A).

Next we will manipulate expression (6) to expose its structure as a sum over a set of conditional probabilities

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{R}\sim p_{\boldsymbol{R}}}\left[\#(\mathcal{S}|\boldsymbol{R})\right] && (7) \\ &= \mathbb{E}_{\boldsymbol{R}\sim p_{\boldsymbol{R}}}\left[\sum_{j=1}^{d} \phi\left(\chi_{\mathcal{S}}^\top \boldsymbol{R}_{:j}\right)\right] && \text{(by definition (5))} \\ &= \sum_{j=1}^{d} \mathbb{E}_{\boldsymbol{R}\sim p_{\boldsymbol{R}}}\left[\phi\left(\chi_{\mathcal{S}}^\top \boldsymbol{R}_{:j}\right)\right] && \text{(sum} \Leftrightarrow \text{expectation)} \\ &= \sum_{j=1}^{d} \underbrace{\Pr\left\{\phi\left(\chi_{\mathcal{S}}^\top \boldsymbol{R}_{:j}\right) = 1 | \chi_{\mathcal{S}}\right\}}_{:=f_j(\chi_{\mathcal{S}})} && (\phi(\cdot) \text{ is binary}), \end{aligned}$$

where $f_j(\chi_{\mathcal{S}})$ is the conditional probability of $\phi\left(\chi_{\mathcal{S}}^\top \boldsymbol{R}_{:j}\right)$ being 1 given the set indicator $\chi_{\mathcal{S}}$.

Strategy for learning: The form of the influence function as a sum over conditional probabilities suggests a simple strategy for learning the influence function:

1. **we learn each $f_j(\chi_\mathcal{S})$ separately,**
2. **and then sum them together,**

which we will elaborate in subsequent sections.

4. Random Basis Function Approximation

In this section, we will provide a novel parameterization of function $f_j(\chi_\mathcal{S})$ using random basis functions. Recall from the derivation in (7) that

$$f_j(\chi_\mathcal{S}) = \mathbb{E}_{r\sim p_j(r)}\left[\phi(\chi_\mathcal{S}^\top r)\right] \tag{8}$$

where $r := \boldsymbol{R}_{:j}$ and $p_j(r)$ is the marginal distribution of column j of $\boldsymbol{R}$ induced by $p_{\boldsymbol{R}}$. Since $f_j(\chi_\mathcal{S})$ is an expectation w.r.t. a distribution $p_j(r)$ over the binary vectors $\{0,1\}^d$, we will use a convex combination of random basis functions to parameterize $f_j(\chi_\mathcal{S})$. A similar idea, called random kitchen sinks (Rahimi & Recht, 2008), has appeared in the classification and kernel methods context. Our use of such parameterization is novel in the information diffusion and coverage function learning context, and our analysis is also different.

Specifically, consider drawing a set of K random binary vectors (random features) $\{r_1, r_2, \ldots, r_K\}$ from some distribution $q(r)$ on $\{0,1\}^n$, and build functions of the form

$$f^w(\chi_\mathcal{S}) = \sum\nolimits_{k=1}^K w_k\,\phi(\chi_\mathcal{S}^\top r_k) = w^\top \boldsymbol{\phi}(\chi_\mathcal{S}), \tag{9}$$

$$\text{subject to } \sum\nolimits_{k=1}^K w_k = 1, w_k \geq 0 \tag{10}$$

where $w := (w_1, \ldots, w_K)^\top$ are parameters to be learned, r_k is the sampled random feature, and $\boldsymbol{\phi}(\chi_\mathcal{S}) := (\phi(\chi_\mathcal{S}^\top r_1), \ldots, \phi(\chi_\mathcal{S}^\top r_K))^\top$. Since each random basis function $\phi(\chi_\mathcal{S}^\top r_k)$ takes value either 0 or 1, the above combination of such functions will qualify as a probability in $[0,1]$. We will denote the class of functions defined by equations (9) and (10) as $\widehat{\mathcal{F}}^w$.

How well can the random basis function $f^w(\chi_\mathcal{S})$ approximate the original function $f_j(\chi_\mathcal{S})$? We can show that there exists some w such that $f^w(\chi_\mathcal{S})$ approximates $f_j(\chi_\mathcal{S})$ well when K is sufficiently large. More specifically, let C be the minimum value such that

$$p_j(r) \leq C q_j(r), \;\; \forall j \in [d], \;\; \forall r \in \{0,1\}^n .$$

Intuitively, C measures how far away the sampling distribution $q_j(r)$ is from the true distribution $p_j(r)$.

Lemma 1. *Let $p_\chi(\chi_\mathcal{S})$ be a distribution of $\chi_\mathcal{S}$. If $K = O(\frac{C^2}{\epsilon^2}\log\frac{C}{\epsilon\delta})$ and $r_1, \ldots, r_K$ are drawn* i.i.d. *from $q_j(r)$, then with probability at least $1-\delta$, there exists an $f^w \in \widehat{\mathcal{F}}^w$ such that $\mathbb{E}_{\chi_\mathcal{S}\sim p_\chi}[(f_j(\chi_\mathcal{S}) - f^w(\chi_\mathcal{S}))^2] \leq \epsilon^2$.*

Alternatively, the lemma can also be interpreted as the approximation error ϵ scales as $O(\frac{C}{\sqrt{K}})$. Note that we require that w lie in a simplex, *i.e.*, $w_k \geq 0$ and $\sum_{k=1}^K w_k = 1$, and it is slightly different from that in Rahimi & Recht (2008).

5. Efficient Learning Algorithm

After generating the random features, we can learn the weights $w = (w_1, \ldots, w_K)$ by fitting $f^w(\chi_\mathcal{S})$ to training data. Since the target function $f_j(\chi_\mathcal{S})$ is a conditional probability, l_2 or l_1 error metric may not be suitable loss functions to optimize. A natural approach is maximum conditional likelihood estimation. We use an efficient exponentiated gradient algorithm for performing the estimation for the weights in f^w. Here, we describe the algorithm, and then present the sample complexity analysis in the next section.

Suppose we observe a dataset of m *i.i.d.* cascades

$$\mathcal{D}^m := \{(\mathcal{S}_1, \mathcal{I}_1), \ldots, (\mathcal{S}_m, \mathcal{I}_m)\}, \tag{11}$$

where each cascade is a pair of observation of the source set $\mathcal{S}_i$ and the corresponding set $\mathcal{I}_i$ of influenced nodes. Each cascade $(\mathcal{S}_i, \mathcal{I}_i)$ in the dataset is obtained by first sampling a source set $\mathcal{S}_i$ from a distribution $p_\chi(\chi_\mathcal{S})$ (*e.g.*, power law), then sampling a random reachability matrix $\boldsymbol{R}$ from $p_{\boldsymbol{R}}$, and finally calculating $\mathcal{I}_i := \left\{j : \phi\left(\chi_\mathcal{S}^\top \boldsymbol{R}_{:j}\right) = 1\right\}$. We note that $\boldsymbol{R}$ is an intermediate quantity which is not observed in the dataset. In our setting, we let $\mathcal{S}_i \subseteq \mathcal{I}_i$ which means the nodes in the source set are also considered as influenced nodes.

For a particular cascade $(\mathcal{S}_i, \mathcal{I}_i)$ and a particular target node j, we can define a binary variable indicating whether the target node j is influenced in this cascade, $y_{ij} := \mathbb{I}\{j \in \mathcal{I}_i\}$. Then the conditional likelihood of the status of node j (influenced or not) can be expressed using $f_j(\chi_\mathcal{S})$

$$f_j(\chi_{\mathcal{S}_i})^{y_{ij}}\,(1 - f_j(\chi_{\mathcal{S}_i}))^{1-y_{ij}}\,. \tag{12}$$

So in the following, we will focus on learning individual function f^w which is an approximation of $f_j(\chi_\mathcal{S})$.

5.1. Maximum conditional likelihood estimation

In a way very similar to logistic regression and conditional random fields by Lafferty et al. (2001), we will maximize the conditional log-likelihood the y_{ij} given the $\chi_{\mathcal{S}_i}$. In contrast to logistic regression and conditional random fields where the models usually take the exponential family form, we will use a form of a convex combination of random basis function (f^w). The additional challenge for this parameterization is that the conditional probability may be zero for some $\mathcal{S}$. To address this challenge, we will use a truncated or Winsorized version of the function f^w

$$f^{w,\lambda}(\chi_\mathcal{S}) = (1-2\lambda)f^w(\chi_\mathcal{S}) + \lambda \tag{13}$$

which squashes the function output to the range of $[\lambda, 1-\lambda]$. We will denote this new class of functions as $\widehat{\mathcal{F}}^{w,\lambda}$. Although this transformation introduces additional bias to the function class, we show in later analysis that it is fine if we choose λ to be about the same level as the approximation error. In practice, λ is selected via cross-validation.

Then the log-likelihood of the data $\mathcal{D}^m$ can be written as

$$\ell_j(w) := \sum_{i=1}^{m} y_{ij} \log f^{w,\lambda}(\chi_{\mathcal{S}_i}) + (1-y_{ij}) \log(1 - f^{w,\lambda}(\chi_{\mathcal{S}_i})), \quad (14)$$

and we can find w by maximizing the log-likelihood

$$\widehat{w} := \underset{w}{\operatorname{argmax}} \quad \ell_j(w) \quad (15)$$

$$\text{subject to} \quad \sum_{k=1}^{K} w_k = 1, w_k \geq 0.$$

One can easily show that the optimization problem in (15) is a convex optimization problem over a probability simplex. Hence we can leverage existing techniques from convex optimization by Kivinen & Warmuth (1997) and Schmidt et al. (2009) to find $\widehat{w}$ efficiently.

5.2. Exponentiated gradient algorithm

We describe a simple exponentiated gradient (EG) algorithm, originally introduced by Kivinen & Warmuth (1997) in the online learning context. The EG updates involve the following simple multiplicative modification

$$w_k^{t+1} = \frac{1}{Z^t} w_k^t \exp\left(-\eta \nabla_k(w^t)\right) \quad (16)$$

where $Z^t = \sum_{k=1}^{K} w_k^t \exp\left(-\eta \nabla_k(w^t)\right)$ is the normalization constant, the parameter $\eta > 0$ is the learning rate, and the gradient $\nabla(w^t)$ is given by

$$\nabla(w) = (1-2\lambda) \sum_{i=1}^{m} \left(\frac{1-y_{ij}}{1-\lambda-(1-2\lambda)w^\top \boldsymbol{\phi}(\chi_{\mathcal{S}_i})} - \frac{y_{ij}}{\lambda + (1-2\lambda)w^\top \boldsymbol{\phi}(\chi_{\mathcal{S}_i})} \right) \boldsymbol{\phi}(\chi_{\mathcal{S}_i}) \quad (17)$$

Algorithm 1 summarizes algorithm for learning the influence function. We first generate K random features $\{r_1, \ldots, r_K\}$ from the given distribution $q_j(r)$. Then, we precompute m feature vectors $\boldsymbol{\phi}(\chi_{\mathcal{S}_i}) = (\phi(\chi_{\mathcal{S}_i}^\top r_1), \ldots, \phi(\chi_{\mathcal{S}_i}^\top r_K))^\top$. Because $\chi_{\mathcal{S}_i}$ is usually very sparse, this preprocessing costs $O(K \sum_{i=1}^{m} |\mathcal{S}_i|)$, where $|\mathcal{S}_i|$ is the cardinality of the set $\mathcal{S}_i$. Then we use the exponentiated gradient algorithm to find the weight w that maximizes the log-likelihood of the training data. According to Kivinen & Warmuth (1997), to get within ϵ of the optimum, we need $O(\frac{1}{\epsilon\eta})$ iterations, where the main work of each iteration is evaluating the gradient with complexity $O(dmK)$. The final estimate $\widehat{\sigma}(\mathcal{S})$ is the sum of all the functions learned for each node. The learning task for each node is independent of those for the other nodes (except that we use the same set of training data), so the algorithm can be easily parallelized. We refer to our algorithm as INFLULEARNER.

5.3. How to choose random basis function

By our analysis in Lemma 1, the number of random features needed for node j depends on the sampling distribution $q_j(r)$. More precisely, it has quadratic dependence on C where $p_j(r) \leq C q_j(r)$ for all r. If we know $p_j(r)$, then by sampling random features from $p_j(r)$, we have $C = 1$ so that much fewer features are needed. However, in practice, $p_j(r)$ is often unknown, so we consider estimating $p_j(r)$ by $q_j(r)$ using the following simple approach.

Algorithm 1 INFLULEARNER

```
input training data $\{(\mathcal{S}_i, \mathcal{I}_i)\}_{i=1}^m$, $\lambda \in (0, \frac{1}{4})$
  for each node $j \in [d]$ do
    sample $K$ random features $\{r_1, \ldots, r_K\}$ from $q_j(r)$;
    compute $\boldsymbol{\phi}(\chi_{\mathcal{S}_i}) = (\phi(\chi_{\mathcal{S}_i}^\top r_1), \ldots, \phi(\chi_{\mathcal{S}_i}^\top r_K)), \forall i$;
    initialize $w^1$ to a interior point of a $K$-simplex;
    for $t = 1, \ldots, T$ do
      calculate $\nabla(w^t)$ using (17)
      update $w^{t+1} \propto w^t \exp(-\eta \nabla(w^t))$ using (16)
    end for
    $\widehat{f}_j^{w,\lambda}(\chi_\mathcal{S}) = \lambda + (1-2\lambda)(w^T)^\top \boldsymbol{\phi}(\chi_\mathcal{S})$
  end for
output $\widehat{\sigma}(\mathcal{S}) = \sum_{j=1}^d \widehat{f}_j^{w,\lambda}(\chi_\mathcal{S})$.
```

Inspired by the empirical success of Naïve Bayes algorithm in classification by Bishop (2006) and the mean field approximation in graphical model inference (Wainwright & Jordan, 2003), we assume that $q_j(r)$ is fully factorized, *i.e.*,

$$q_j(r) = \prod_{s=1}^{d} q_j(r(s)).$$

where $q_j(r(s))$ means the marginal distribution of the i-th dimension of r. Given a training dataset $\mathcal{D}^m$ as in equation (11), we estimate each $q_j(r(s))$ using the frequency of node j being influenced by source node i, *i.e.*, $q_j(r(s)) = \frac{1}{|\mathcal{D}_s^m|} \sum_{i \in \mathcal{D}_s^m} y_{ij}$ where $\mathcal{D}_s^m := \{i : s \in \mathcal{S}_i\}$. Although this $q_j(r)$ may be quite different from $p_j(r)$, by the additional steps of drawing random features and adjusting the corresponding weights, it leads to very good results, as illustrated in our experiments.

A more intelligent approach for choosing $q_j(r)$ may be first learning a diffusion model outlined in Section 2 and then using samples from the diffusion model to generate the random basis functions. This approach requires more computation and is left for future study.

6. Sample Complexity of MLE

Here we analyze Algorithm 1 and provide sample complexity bounds for the number of random basis functions and the size of the training data needed to get a solution close to the truth. We describe our results here and provide the proof in the appendix.

We note that existing analysis for random kitchen sink (Rahimi & Recht, 2008) does not apply to the maximum likelihood estimation. Therefore, we use a general framework by Birgé & Massart (1998) for maximum likelihood estimation. Loosely speaking, the error of the maximum likelihood estimator $\widehat{f}_j^{w,\lambda}(\chi_{\mathcal{S}}) \in \widehat{\mathcal{F}}^{w,\lambda}$ is bounded by the best possible in the hypothesis class plus a term scale roughly as $\tilde{O}(D/m)$, where D is the dimension of the set of candidate models based on a covering approach. Hence, to get sample complexity bounds for our problem, we need to bound the dimension of $\widehat{\mathcal{F}}^{w,\lambda}$. We consider the mapping from the weight w to the corresponding hypothesis $f \in \widehat{\mathcal{F}}^{w,\lambda}$, and show that the distance between two functions f and f' are approximately the distance between their corresponding weights w and w'. Then a covering on the space of w induces a covering on the function space $\widehat{\mathcal{F}}^{w,\lambda}$, and thus the dimensions of the two spaces are approximately the same, which is $O(K)$. Combined the dimension bound with Lemma 1, we arrive at the following:

Lemma 2. *Assume the statement in Lemma 1 is true. If $m = \tilde{O}(\frac{K}{\epsilon})$, then the maximum likelihood estimator $\widehat{f}_j^{w,\lambda} \in \widehat{\mathcal{F}}^{w,\lambda}$ satisfies*

$$\mathbb{E}_{\mathcal{D}^m}\mathbb{E}_{p_{\mathcal{X}}}\left[(\widehat{f}_j^{w,\lambda}(\chi_{\mathcal{S}}) - f_j(\chi_{\mathcal{S}}))^2\right] \leq \tilde{O}\left(\frac{\epsilon^2 + \lambda^2}{\lambda}\right).$$

This means to get ϵ accuracy, it suffices to choose $\lambda = \epsilon$ and choose K large enough to make sure that the l_2 error between the true function and the set of candidate functions in $\widehat{\mathcal{F}}^{w,\lambda}$ is at most ϵ^2. The bound then follows by applying the above argument on each node with accuracy $O(\epsilon/d)$.

Theorem 3. *Suppose in Algorithm 1, we set $\lambda = \tilde{O}(\frac{\epsilon}{d})$, $K = \tilde{O}(\frac{C^2 d^2}{\epsilon^2})$, and $m = \tilde{O}\left(\frac{C^2 d^3}{\epsilon^3}\right)$. Then with probability at least $1 - \delta$ over the drawing of the random features, the output of Algorithm 1 satisfies*

$$\mathbb{E}_{\mathcal{D}^m}\mathbb{E}_{p_{\mathcal{X}}}\left[\left(\sum_{j=1}^d \widehat{f}_j^{w,\lambda}(\chi_{\mathcal{S}}) - \sigma(\mathcal{S})\right)^2\right] \leq \epsilon.$$

Intuitively, the l_2 error of the function $\sum_{j=1}^d \widehat{f}_j^{w,\lambda}$ learned is small if the number K of random features and the size m of the training data are sufficiently large. Both quantities have a quadratic dependence on C, since if C is large, then the difference between p_j and q_j could be large, and thus we need more random features to approximate f_j and also more training data to learn the weights. K and m also depend on the number d of nodes in the network, for the reason that we need to estimate each f_j up to accuracy ϵ/d so that their sum is estimated to accuracy ϵ. This is far too pessimistic, as we observe in our experiment that much smaller K or m is needed.

7. Experiments

We evaluate InfluLearner in synthetic and real world data. We compare it to the state-of-the-art two-stage approaches, as well as methods based on linear regression and logistic regression, and show that InfluLearner is more robust to model misspecification than these alternatives.

7.1. Competitors

Two-stage methods. Two-stage learning methods depend on the diffusion model assumptions, families of pairwise temporal dynamics, and whether network structures are given or not. We design the following four representative competitors :

1. **C**ontinuous-time **I**ndependent **C**ascade model with exponential pairwise transmission function (CIC).
2. **C**ontinuous-time **I**ndependent **C**ascade model with exponential pairwise transmission function and given network **S**tructure (CIC-S).
3. **D**iscrete-time **I**ndependent **C**ascade model (DIC).
4. **D**iscrete-time **I**ndependent **C**ascade model with given network **S**tructure (DIC-S).

For the methods CIC and CIC-S, we use NETRATE (Gomez Rodriguez et al., 2011) to learn the structure and parameters of the pairwise transmission functions. For DIC and DIC-S, we learn the pairwise infection probability based on the method of (Netrapalli & Sanghavi, 2012).

Approach based on logistic regression. Instead of using random features, we represent $f_j(\chi_{\mathcal{S}})$ using a modified logistic regression

$$f_j(\chi_{\mathcal{S}}) = \frac{2\exp\left(w^\top \chi_{\mathcal{S}}\right)}{1 + \exp\left(w^\top \chi_{\mathcal{S}}\right)} - 1, \text{ where } w \geq 0. \quad (18)$$

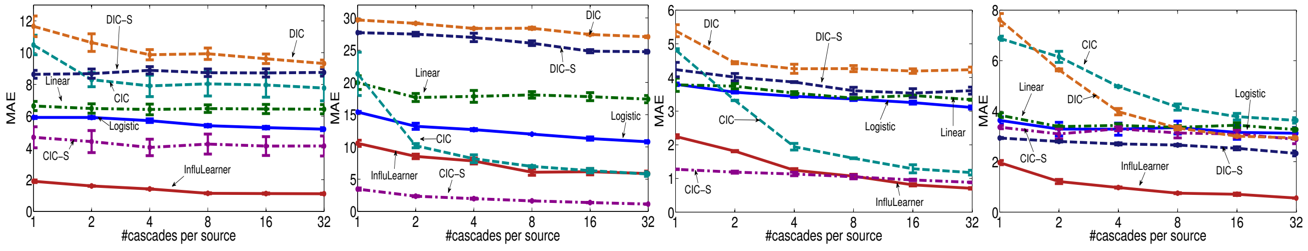
Since the sigmoid function is concave in $\mathbb{R}_+$ and $w^\top \chi_{\mathcal{S}}$ is a linear function of $\chi_{\mathcal{S}}$, the representation in (18) is also a submodular function of the set $\mathcal{S}$. We learn w by maximizing the log-likelihood subject to the nonnegative constraint. We also experimented with the original logistic regression model which does not lead to a submodular function, and thus does not perform as well as the representation in (18) (and hence not reported).

Approach based on linear regression. We use the linear regression model, $w^\top \chi_{\mathcal{S}} + b$, to directly regress from $\chi_{\mathcal{S}}$ to the cascade size $|\mathcal{I}|$. This approach does use the knowledge that the influence function is a coverage function.

7.2. Synthetic Data

We generate Kronecker type of synthetic networks with the parameter matrix [0.9 0.5; 0.5 0.3], which mimics the information diffusion traces in real world networks (Leskovec et al., 2010). The generated networks consist of 1,024 nodes and 2,048 edges. Given a generated network structure, we apply the continuous-time independent cascade, the discrete-time independent cascades and the linear-threshold model to generate the cascades, respectively.

For the continuous-time diffusion model, we used both Weibull distribution (Wbl) and exponential distribution (Exp) for the pairwise transmission function, and set their parameters at random to capture the heterogeneous temporal dynamics. For the Weibull distribution, $f(t; \alpha, \beta) =$



(a) Weibull Family (CIC) (b) Exponential (CIC) (c) DIC (d) LT

Figure 1. Over the generated synthetic networks with 1,024 nodes and 2,048 edges, we present the mean absolute error of the estimated influence on the testing data by increasing the number of training data when the true diffusion model is (a) continuous-time independent cascade with pairwise Weibull transmission functions, (b) continuous-time independent cascade with pairwise exponential transmission functions, (c) discrete-time independent cascade model and (d) linear-threshold cascade model.

$\frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} e^{-(t/\alpha)^{\beta}}, t \geqslant 0$, where $\alpha > 0$ is a scale parameter and $\beta > 0$ is a shape parameter. We choose α and β from 1 to 10 uniformly at random for each edge in order to have heterogeneous temporal dynamics. The true influence value range is from 1 to 235, and the average value is 15.78 with the time window $T = 10$. For the exponential distribution, the average influence is 37.81.

For the discrete-time independent cascade model, the pairwise infection probability is chosen uniformly from 0 to 1. For the discrete-time linear-threshold model, we followed Kempe et al. (2003) where the edge weight w_{uv} between u and v is $1/d_v$, and d_v is the degree of node v. We run these generative models for 10 time steps. The average influence values are 9.2 and 8.9 respectively.

The source locations are sampled uniformly without replacement from $\mathcal{V}$, and the source set sizes conform to a power law distribution with parameter 2.5. For the training set, we independently sample 1,024 source sets, and independently generate 8 to 128 cascades for each source set. The test set contains 128 independently sampled source sets with the ground truth influence estimated from 10,000 simulated cascades.

7.3. Robustness to model misspecification

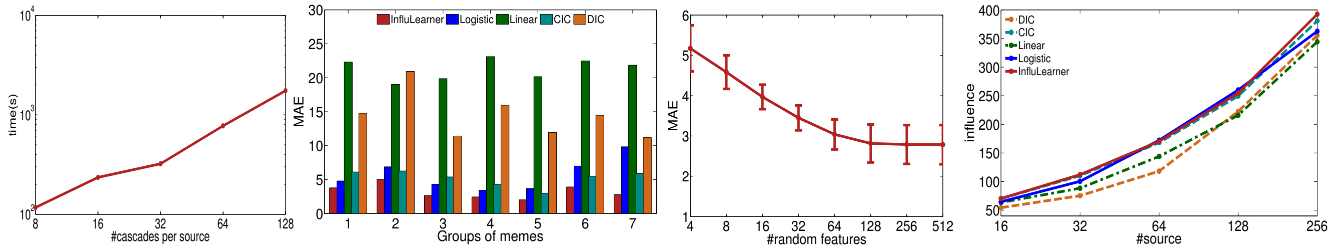
The cascades used in Figure 1(a) are generated from the continuous-time independent cascade model with pairwise Weibull transmission functions. We expect that the four two-stage methods are not doing well due to model misspecification of one form or the other. Figure 1(a) shows the MAE (Mean Absolute Error) between the estimated value and the true value. Both CIC-S and CIC used the correct continuous-time diffusion model but the wrong family of pairwise transmission functions, so their performance lies in the middle. However, CIC-S has the prior knowledge about the true network structure, so it is reduced to a much simpler learning problem and is thus better than CIC. DIC-S and DIC used the wrong diffusion model with unit time step (which is hard to determine in practice), so they have the lowest performance overall. In contrast, INFLULEARNER does not explicitly make assumptions about diffusion models or transmission functions but only learns the influence function directly from the data. Thus, it is much more robust and better than the two-stage methods. Since INFLULEARNER has better representational power than the logistic regression based approach, it is able to better approximate the true influence function and thus can achieve the best performance overall.

The cascades used in Figure 1(b) are generated from the continuous-time independent cascade model with pairwise exponential transmission functions. Note that in this case we expect CIC-S and CIC to do well, since they have the correct assumptions about both the diffusion model and the family of transmission functions. Particularly, with the prior knowledge of the true network structure, CIC-S simply fits the model parameters for each edge, and thus the estimates converge to the true influence function quickly. Still, we see that the performance of INFLULEARNER is close to that of CIC-S and CIC. Figure 1(b) again show that INFLULEARNER is robust to diffusion model changes.

In Figure 1(c, d), we generate cascades according to discrete-time independent cascade model and linear threshold model respectively. In Figure 1(c), DIC-S and DIC assumes the correct model, so their performance improves a lot. However, in Figure 1(d), because CIC-S, CIC, DIC-S, and DIC all assume the wrong diffusion model, we observe a similar trend as as in Figure 1(a): INFLULEARNER is robust and obtain the best results. Note that in this case, the gap between different methods is not as big since the average influence value is small.

7.4. Scalability

Figure 2(a) reports the parallel runtime of INFLULEARNER as we increase the number of training cascades per source set. We arbitrarily divide the 1,024 independent learning problems into 32 individual jobs running on a cluster of 32 cores (AMD Opteron(tm) Processor, 2.5GHz). It shows that the runtime grows almost linearly as the number of cascades increases.



(a) Runtime (b) MAE on real data (c) Effect of random features (d) Influence maximization

Figure 2. (a) Runtime in log-log scale; (b) MAE on seven sets of real cascade data; (c) The performance gain of using different number of random features; (d) Maximized expected influence of different selected sources on the real hold-out testing data.

7.5. Influence estimation on real data

We further evaluate the performance of our proposed method on the MemeTracker dataset which includes 300 million blog posts and articles collected from 5,000 active media sites between March 2011 and February 2012 (Leskovec et al., 2009). The flow of information was traced using quotes which are short textual phrases spreading through the websites. Because all published documents containing a particular quote are time-stamped, a cascade induced by the same quote like 'apple and jobs' is a collection of times when the media site first mentioned it. We have selected seven groups of cascades with the typical keywords like 'apple and jobs', 'tsunami earthquake', 'william kate marriage', 'occupy wall-street', 'airstrikes', 'egypt' and 'elections'. We split each set of cascades into 60%-train and 40%-test. Because we do not have any prior knowledge about either the diffusion structure or the underlying diffusion mechanism on the real cascades data, we only compare INFLULEARNER with the Logistic regression, Linear regression, CIC and DIC.

We evaluate the performance on the held-out testing cascades as follows : we randomly select 10 sources from the testing cascades, which represents one particular source set S. For each node $u \in S$, let $\mathcal{C}(u)$ denote the set of cascades generated from u on the testing data. For each $u \in S$, we uniformly sample one cascade from $\mathcal{C}(u)$. Thus, the union of all sampled cascades is the set of nodes infected by source set S. We repeat the process for 1,000 times and take the average of the number of infected nodes as the true influence of source set S. Finally, we have generated 100 source sets and report the MAE of each method in Figure 2(b). We can see that the performance of INFLULEARNER is robust and consistent across all groups of testing cascades, and is significantly better than the other competitors.

Moreover, Figure 2(c) demonstrates the effect of the number of random features on the performance of INFLULEARNER by showing the average MAE over the seven sets of cascade data as the number of random features increases. As the number of random features grows, INFLULEARNER approximates the true influence better, and thus the MAE decreases. It seems that 128 to 256 random features are sufficient to achieve good performance overall.

7.6. Influence maximization on real data

Finally, we use the learned influence function (from INFLULEARNER, Logistic, Linear, CIC and DIC) for solving the influence maximization problem Kempe et al. (2003); Du et al. (2013b). Here we want to find a set $\mathcal{S}^*$ of C source nodes which maximizes the influence, *i.e.*, $\mathcal{S}^* = \operatorname{argmax}_{|\mathcal{S}|\leq C} \sigma(\mathcal{S})$. We will use a greedy algorithm framework of Nemhauser et al. (1978) to solve the problem. We use the held-out test cascade to estimate the influence achieved by selected source nodes. The observation time window used is $T = 14$.

Figure 2(d) shows the influence achieved in Meme group 1 (the rest of the testing groups has similar results as in the Appendix). INFLULEARNER, Logistic and CIC perform consistently better than DIC and linear regression. The source nodes selected by INFLULEARNER, Logistic and CIC are very similar, though the estimated influence value can be different. As a result, the influence value of INFLULEARNER, Logistic and CIC are very close.

8. Conclusion

Based on the observation that the influence function in many diffusion models are coverage functions, we propose to directly learn the influence from cascade data. In this paper, we provide a novel parameterization of the influence function as a convex combination of random basis functions, and an efficient maximum likelihood based algorithm for learning the weighting of the random basis functions. Theoretically, we show that the algorithm can learn the influence with low sample complexity, and our empirical study also shows our method outperforms traditional two-stage approaches.

Acknowledgement

The research was supported in part by NSF/NIH BIGDATA 1R01GM108341, NSF IIS-1116886, NSF CAREER IIS-1350983 to L.S.; NSF CCF-1101283, NSF CAREER CCF-0953192, AFOSR FA9550-09-1-0538, ONR N00014-09-1-0751, and a Microsoft Research Faculty Fellowship to M.B; a Raytheon Faculty Fellowship to M.B. and L.S.; and a Facebook Fellowship to N .D.

References

Badanidiyuru, A., Dobzinski, S., Fu, H., Kleinberg, R. D., Nisan, N., and Roughgarden, T. Sketching valuation functions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2012.

Balcan, Maria-Florina and Harvey, Nicholas JA. Learning submodular functions. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pp. 793–802. ACM, 2011.

Birgé, L. and Massart, P. Minimum Contrast Estimators on Sieves: Exponential Bounds and Rates of Convergence. *Bernoulli*, 4(3), 1998.

Bishop, Christopher. *Pattern Recognition and Machine Learning*. Springer, 2006.

Borgs, Christian, Brautbar, Michael, Chayes, Jennifer, and Lucier, Brendan. Influence maximization in social networks: Towards an optimal algorithmic solution. *arXiv preprint arXiv:1212.0884*, 2012.

Chen, Wei, Wang, Chi, and Wang, Yajun. 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.

Du, N., Song, L., Smola, A., and Yuan, M. Learning networks of heterogeneous influence. In *Advances in Neural Information Processing Systems 25*, pp. 2789–2797, 2012.

Du, N., Song, L., Woo, H., and Zha, H. Uncover topic-sensitive information diffusion networks. In *Artificial Intelligence and Statistics (AISTATS)*, 2013a.

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

Feldman, Vitaly and Kothari, Pravesh. Learning coverage functions. *arXiv preprint arXiv:1304.2079*, 2013.

Feldman, Vitaly and Vondrak, Jan. Optimal bounds on approximation of submodular and xos functions by juntas. In *FOCS*, 2013.

Gomez Rodriguez, Manuel, Balduzzi, David, and Schölkopf, Bernhard. Uncovering the temporal dynamics of diffusion networks. *arXiv preprint arXiv:1105.0697*, 2011.

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.

Kivinen, J. and Warmuth, M. K. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

Lafferty, J. D., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *Proceedings of International Conference on Machine Learning*, volume 18, pp. 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

Leskovec, Jure, Backstrom, Lars, and Kleinberg, Jon. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 497–506. ACM, 2009.

Leskovec, Jure, Chakrabarti, Deepayan, Kleinberg, Jon, Faloutsos, Christos, and Ghahramani, Zoubin. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research*, 11(Feb):985–1042, 2010.

Nemhauser, G., Wolsey, L., and Fisher, M. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

Netrapalli, Praneeth and Sanghavi, Sujay. Learning the graph of epidemic cascades. In *SIGMETRICS/PERFORMANCE*, pp. 211–222. ACM, 2012. ISBN 978-1-4503-1097-0.

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

Rodriguez, M.G. and Schölkopf, B. Influence maximization in continuous time diffusion networks. In *Proceedings of the International Conference on Machine Learning*, 2012.

Schmidt, M., van den Berg, E., Friedlander, M. P., and Murphy, K. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. In van Dyk, D. and Welling, M. (eds.), *Proceedings of The Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS) 2009*, volume 5, pp. 456–463, Clearwater Beach, Florida, April 2009.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Department of Statistics, September 2003.