Towards large-scale social networks with online diffusion provenance detection

Haishuai Wang, Jia Wu*, Shirui Pan, Peng Zhang, Ling Chen

Centre for Quantum Computation and Intelligent Systems (QCIS), Faculty of Engineering & Information Technology, University of Technology Sydney, NSW 2007, Australia

A R T I C L E   I N F O

Article history:
Received 28 January 2016
Revised 11 August 2016
Accepted 28 August 2016
Available online 30 August 2016

Keywords:
Source locating
Social network
Online identification
\( \ell_1 \) Regression

A B S T R A C T

In this paper we study a new problem of online discovering diffusion provenances in large networks. Existing work on network diffusion provenance identification focuses on offline learning where data collected from network detectors are static and a snapshot of the network is available before learning. However, an offline learning model does not meet the need for early warning, real-time awareness, or a real-time response to malicious information spreading in networks. To this end, we propose an online regression model for real-time diffusion provenance identification. Specifically, we first use offline collected network cascades to infer the edge transmission weights, and then use an online \( \ell_1 \) non-convex regression model as the identification model. The proposed methods are empirically evaluated on both synthetic and real-world networks. Experimental results demonstrate the effectiveness of the proposed model.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

In recent years, information diffusion in large networks has attracted much attention. The spread of malicious information such as viruses, spams and rumors has made various networks vulnerable to privacy attacks, viral advertising, etc. To stop the propagation of malicious information, researchers have recently proposed several models to identify the diffusion provenances in large networks. Online diffusion provenance discovery is also a significant presence on social networks in business. No matter how small, medium or large a business is, a brand’s health and reputation is often defined by the information diffused in social media. While fake reviews or deceptive messages, spread by malevolent people, in social media are inevitable, they may cause damage to a brand’s reputation or corporation’s reputation as a whole. A recent study, Spam Trends in Today’s Business World [1], reported that the productivity cost of malicious information to European companies was an estimated US$2.8 billion, while US-based companies reported a loss of US$20 billion. According to The Washington Post,\(^1\) on Tuesday, April 24, 2013, a single hoax message sent via Twitter erased $200 billion from the US stock market in 2 minutes. In cases like these, locating and severing the provenances of the diffusion in a timely manner is critical.

A false rumor (i.e., a libelous statement) about the financial performance of a firm may be spread by market manipulators to influence the price of the firm’s stock, resulting in fines from regulators or data protection enforcement agencies. While legal recourse exists for victims of libel, law enforcement agencies still need to identify the original source of the rumor and those who spread it. If the perpetrators are anonymous, tracing the IP address and identities of the individual profiles carrying or linking to the opinion piece becomes significantly more difficult and time-consuming.

Existing diffusion provenance identification models can be roughly categorized into two classes: snapshot-based provenance identification [2,3] and detector-based identification [4–6]. Snapshot-based methods operate on the assumption that a snapshot of the entire network can be obtained and provenances can be estimated under stochastic propagation models such as the SI [7] and SIR [8] models. Although these methods have shown promising results in experiments, fetching a snapshot of the entire network is very expensive, if not impossible. Detector-based methods assume that only a small subset of nodes in a network can be monitored, and provenances can be inferred from the observations (samples) of these detectors. This group of methods has recently attracted increasing attention due to its potential for use in real-world applications.

However, to the best of our knowledge, most existing work on the diffusion provenance locating problem falls into the category

Motivated by the urgent demand for real-time and continuous diffusion provenance detection in networks, we propose to use regression learning as the basic detection model. Regression learning is favorable for real-time applications because of the freely available prior distribution. Fig. 2 shows a network propagation with only one provenance $S_0$. We extract a propagation path with seven nodes $\{S_0, \ldots, S_7\}$. Assume that we are able to observe nodes $S_2$ and $S_4$ (detectors). The two nodes are activated at time points $t^* + \tau$ and $t^* + 2\tau$ respectively. The goal is to use the least square regression to minimize the error rate between the observed time delay and the estimated time delay. Assume at time $t_i \in T = [t^*, t^* + 3\tau]$, the $i$th node is observed. At any time $t_i$, we have, $\min \sum_{j=2,4} \{ |(t^* + i\tau) - a_{ij}^2x|^2 \}$. s.t.: $x \geq 0$, where $x \in \mathbb{R}^7$ is the target variable with element $x_i$ denoting the probability that node $i$ is the diffusion provenance. $a_{ij} \in \mathbb{R}^7$ is a column vector with element $a_{ij}$ denoting the propagation path length from the detector $i$ to the $j$th node, e.g., $a_2 = (1, 2, 0, 1, 3, 4, 2)^T$ and $a_4 = (2, 1, 3, 4, 0, 1, 5)^T$. The objective function is convex and non-negative and achieves its minimum value 0 when $x = (1, 0, 0, 0, 0, 0, 0)^T$. In offline detection, since data from both nodes $S_2$ and $S_4$ are known, we can obtain the result as $x = (1, 0, 0, 0, 0, 0, 0)^T$, which correctly indicates that $S_0$ is the diffusion provenance. However, in online detection, we first estimate $x$ by observing data only from detector $S_2$ and the result is $\{0, 0.8, 0, 0.2, 0, 0, 0\}^T$. When data from $S_4$ arrive, the result of $x$ is updated to $\{1, 0, 0, 0, 0, 0, 0\}^T$. We can see that an online algorithm demands dynamic and continuous computation of $x$, and the result of online learning is expected to approximate (or equal) the offline result.

Compared to offline identification methods, online identification models have the following challenges:

- **Challenge 1**: how to design an online identification model? The online identification model needs to address five questions, the unknown number of diffusion provenance $k$, both activated and inactivated detectors, the unknown initial propagation time $t^*$, the uncertain propagation path, and the uncertain propagation time delay. The unknown number of diffusion provenance $k$ indicates a sparse solution of $x$. Both activated and inactivated detectors lead to partially labeled data and a non-convex objective function. The uncertain propagation path and propagation time delay demand an aggregate Gaussian distribution to describe the estimated propagation time delay.

- **Challenge 2**: how to design a stochastic learning algorithm to solve the online identification model? Because the detectors are activated sequentially, and a decision needs to be made once a detector is activated, the algorithm needs to digest data in a continuous and converging way. Ideally, the results of the online algorithm approximate those of the offline algorithms.

- **Challenge 3**: how to evaluate the performance of the proposed online identification method? Given the unique characteristics of
the problem, various data are demanded to evaluate performance.

In this paper, we propose a new online regression learning model to identify diffusion provenances in large networks. To solve Challenge 1, we use an $l_1$ non-convex regression learning model built upon an aggregate Gaussian propagation time delay, where the network transmission weights are inferred a priori from offline collected cascades. To tackle Challenge 2, we present an Online Stochastic Sub-gradient (OSS) algorithm that can converge to local minima. Also, we evaluate the model using four synthetic network data to address Challenge 3.

The contributions of the work are twofold:

- We present a new online regression learning model to identify diffusion provenances in large networks. The proposed model can handle the issues of the unknown number of diffusion provenances $k$, the partially activated detectors, the unknown initial propagation time $t^*$, the uncertain propagation path, and the uncertain propagation time delay.
- We present an online stochastic algorithm to solve the proposed online regression learning model. The algorithm uses a stochastic sub-gradient descent algorithm to continuously detect the provenances.

The remainder of the paper is organized as follows. Section 2 surveys related work. Section 3 introduces the related preliminaries. The regression learning and online algorithm are given in Sections 4 and 5 respectively. Section 6 empirically evaluates the algorithms. We conclude the paper in Section 7.

2. Related work

Malicious information, such as rumors and viruses, that propagate in networks, which result in privacy and security concerns [9,10] and motivates this research into diffusion provenance detection. To date, existing works on diffusion provenance detection have focused on offline detection, where a snapshot of a large network or data harvested from detectors is assumed to be available in advance. In order to design an online detection algorithm, three technical questions need to be answered: 1) how to design stochastic propagation models, 2) how to design an objective function for online detection, and 3) how to design an online algorithm as the solution. We surveyed related work in the three contexts.

In terms of stochastic propagation models for diffusion provenance location, existing works simulate the spread by using infection models such as the Susceptible-Infected-Recovered (SIR) [8] model, the Susceptible-Infected SI [7] model, and others [11,12]. Conversely, the recent works [13-15] have claimed that modeling propagation cascades and information diffusion using continuous-time diffusion networks can provide accurate models.

Several learning models based on the stochastic models were proposed to infer provenances. For example, a recent work [7] provided a systematic survey for locating the provenance of rumor in a network, and presented a rumor centrality estimator to estimate the rumor provenances by assigning a score to each infected node. The work [16] studied the problem of a single rumor provenance locating with a priori knowledge. Most existing estimators are based on either topological centrality measures [17] or distance measures between the observed data. Then, a maximum likelihood estimator can be used to infer the provenances.

One limitation of the above works is that they all assume that the infection status of the nodes (i.e., labels) is known beforehand. For example, the works [4,7] considered the multiple infection provenances estimation problem and assumed that the number of infection provenances is unknown in advance. Some work [6] assumed that not all nodes are infected and only a subset of detectors are used for the provenance estimation.

In online algorithms, online learning has been extensively studied in machine learning. Typical methods include the Passive-Aggressive (PA) [18] and truncated gradient algorithms [19]. However, these online learning algorithms are based on linear and convex optimization, which do not fit our non-convex online learning problem.

Despite the complexity of inferring the diffusion provenances in a network, a simple heuristic is to say that the provenance is the center of the network [20]. There are many notions of network centrality, but a very common one is known as distance centrality, e.g., betweenness centrality [21], closeness centrality [22] and Bonacich centrality [23]. Betweenness centrality measures a node’s centrality in a network. The infection closeness centrality heuristic claims the node with the maximum infection closeness is the source. Bonacich Centrality is a measure of the influence of a node in a network. One may argue that the most influential nodes are more likely to be the provenances of the diffusion. However, actually, an almost isolated node that has few connections to the most influential nodes is probably the source. Therefore, traditional central-based algorithms are hardly applicable.

To sum up, none of the aforementioned works can be directly used to address the online diffusion provenance detection problem studied in this work.

3. Preliminaries

Consider a network $G = (V,E)$, where the vertex set $V$ has $N$ nodes, and the edge set $E$ has $L$ edges. In the network, we have two types of data for model training:

1. **Offline data of cascades**: a set $C$ of cascades $\{c_1, \cdots, c_n\}$, where each cascade $c_i$ is a sequence of activated times $\{t_1, \cdots, t_k\}$ within a given time window $T$. Each time point $t_i$ records the $i$th activated node. For nodes that are not activated within window $T$, the activated time is unknown.

2. **Online data collected from detectors**: We budget a small subset of detectors, denoted by $D = \{d_1, \cdots, d_m\}$, to monitor the network. During the monitoring time window $[t^*, T^* + T]$, where $t^*$ is the initial propagation time and $T$ is the size of the window, there are a subset of detectors activated, denoted by $D_a$, and the remaining inactivated detectors are denoted by $D_r$. We aim to estimate the locations of provenances, denoted by a random vector $s^* \in R^N$, given the status of all the detectors $D = \{(d_1, t_1 - t^*), \cdots, (d_m, t_m - t^*)\}$, where $t_m$ is the activated time point of detector $d_m$ and label $t_m - t^*$ denotes the time delay. Note that time labels of inactivated nodes are unknown, and their time delay exceeds window $T$.

We adopt an SI propagation model to describe how the infection spreads in network $G$. The reasons for using this kind of propagation model are that most posts on social networks are usually not removed, i.e., an infected node stays infected. Thus, SI is an appropriate model for online postings and modeling opinion dynamics on social networks. In the SI model, each node in a network has three possible states: susceptible $(s)$, infected $(i)$ and non-susceptible $(n)$. As the infected nodes are those nodes that possess the infection, and will remain infected throughout, an infected detector cannot be recovered. Therefore, a detector cannot receive the same information multiple times and will not send the same information multiple times to the same node.

Now we infer the edge transmission probability matrix $W$ based on the cascades collected offline. Given a cascade $c_i \in C$, we use $p(c_i|W)$ to denote the likelihood function of observing the cascade $c_i$ under an unknown transmission matrix $W$. The likelihood function consists of the joint probability of activated nodes $v_i \in V (t_i \leq T)$ and inactivated nodes $v_m \in V (t_m > T)$. For an activated
node $v_i \in V$ activated at time $t_i \leq T$ from an adjacent node $v_j \in V$ and $t_j < t_i$, the probability of observing $v_i$ activated by $v_j$ at time $t_i$ is a joint probability of $v_j$ infecting $v_i$ at time $t_j$ and $v_i$ is not activated by any other neighbor $v_k$ which has already been activated by the time $t_i$, i.e., $t_k < t_i$. By summing up all the possible neighbors $v_j$ in the network, i.e., summing up all $v_j$ with $t_j < t_i$, we can achieve the probability $f(c_i | W)$ of a node $v_i$ activated at time $t_i$ as in Eq. (1).

$$f(c_i | W) = \sum_{v_j \in g(v_i) \atop t_j < t_i} \left[ P(t_j \rightarrow t_i | W) \prod_{v_k \in g(v_i) \atop t_k \neq t_j, t_k < t_i} P(t_k \rightarrow t_i | W) \right] .$$

(1)

where $g(v_i)$ denotes the set of neighbors to $v_i$, and $P(t_j \rightarrow t_i | W)$ denotes the probability of the $j$th activated node $v_j$ activates its neighbor $v_i$ at time $t_i$. We can use three well-known parametric functions for $P(t_j \rightarrow t_i | W)$, such as the widely used exponential, power-law and Rayleigh models [13]. Without loss of generality, we use the exponential model, where $P(t_j \rightarrow t_i | W) = \lambda e^{-\lambda t_i}$ if $t_j < t_i$ and 0 otherwise. The probability $P(t_j \rightarrow t_i | W) = 1 - P(t_j \rightarrow t_i | W)$. Based on Eq. (1), the probability of observing all the activated nodes in a cascade $c_i$ is as follows,

$$f(c_i | W) = \prod_{v_i \in s \subseteq T} f(c_i | W) .$$

(2)

Because inactivated nodes also provide information for transmission weight estimation, the probability of observing an entire cascade $c_i$, $f(c_i | W)$, is the joint probability of observing all the activated nodes in the cascade $c_i$ and discarding the remaining inactivated nodes as in Eq. (3).

$$f(c_i | W) = f(c_i | s) \times \left( \prod_{v_i \in s \subseteq T} P(t_i \rightarrow t_m | W) \right) .$$

(3)

The likelihood function of observing all cascades $C$ is the product of all the likelihoods of each cascade given in Eq. (3).

$$L(W) = \log \prod_{c_i \in \mathcal{C}} f(c_i | W) = \sum_{c_i \in \mathcal{C}} \log f(c_i | W) .$$

(4)

Then, the edge transmission matrix $W$ can be estimated as follows,

$$W^* = \arg \max_{w \in \mathbb{R}} L(W) , \text{ s.t. } W \succeq 0 .$$

(5)

For simplicity, we denote $\phi(W; j, i) = P(t_j \rightarrow t_i | W)$, then Eq. (4) can be rewritten as in Eq. (6).

$$L(W) = \sum_{c_i \in \mathcal{C}} \left( \sum_{v_j \in g(v_i) \atop t_j \leq T} \log \sum_{v_j \in g(v_i) \atop t_j < t_i} \frac{\phi(W; j, i)}{1 - \phi(W; j, i)} + \sum_{v_i \in \mathcal{V} \atop t_i \leq T} \sum_{v_k \in g(v_i) \atop t_k < t_i} \log [1 - \phi(W; j, i)] + \sum_{v_i \in \mathcal{V} \atop t_i \leq T} \sum_{v_m \in \mathcal{V} \atop t_m > T} \log [1 - \phi(W; i, m)] \right) .$$

(6)

The likelihood in Eq. (6) is convex, and thus the first-order gradient guarantees a global optimum. By letting the derivative of Eq. (6) to 0, we obtain $\phi'(W_{ji}) = 0$ for each cascade. That is, $W_{ji} = \frac{1}{t_i - t_j}$ for each observed pair of neighboring nodes. As we don’t observe the exact time of $t_m$, for each inactivated node $t_m > T$ in a cascade, we approximately let $t_m = T$ and $W_{jm} = \frac{1}{t_m - t_j}$. Moreover, we set the default propagation probability of each edge to be $W_{ji} = \frac{1}{t_i - t_j}$ in case we don’t observe any propagation data between nodes $v_j$ and $v_i$ in a cascade. Then, by averaging over $|\mathcal{C}|$ cascades, we obtain the final result as follows,

$$W^*_{ji} \propto \frac{1}{k} \sum_{c_i \in \mathcal{C}} l_i(t_i, v_i) \frac{1}{t_i - t_j} , \quad t_i, t_j < T ; \quad t_i > T , t_j \leq T .$$

(7)

where $l_i(t_i, v_i)$ is an indicator and equals to 1 if the cascade $c_i$ satisfies the time constraint. $k$ is the total number of node pairs that meet the constraint. If the given time constraints are not met, the default weight $W_{ji}$ is $1/T$.

4. Regression model

In this section, we formulate the objective function for diffusion provenances detection based on online data collected from the detectors. We assume that the prior distribution of $s^*$ is uniform over the network, i.e., any node in the network is equally likely to be the provenance. Thus, the location of the provenances can be recovered by maximizing the likelihood function of the observed status of the detectors $D$ given provenances $s \in \mathcal{G}$, as shown in Eq. (8),

$$s^* = \arg \max_{s \in \mathcal{G}} P(D | s) = \arg \max_{s \in \mathcal{G}} P(D | s) \left[ 1 - P(D | s) \right] .$$

(8)

where $P(D | s)$ denotes the probability of observing $D$ given a set of spread provenances $s$, and $1 - P(D | s)$ denotes the probability that $D$ is inactivated under provenances $s$. Thus, $P(D | s) \left[ 1 - P(D | s) \right]$ denotes the joint probability of both observing the activated nodes $D$ and inactivated nodes $D_0$. Because malicious information is often spread under the snowball phenomenon [24], it is reasonable to approximate the shortest-path spread, denoted by $P(i, j)$ between nodes $v_i, v_j \in V$.

Because the offline obtained edge transmission matrix $W$ may suffer unstable change when used online, we further use a random variable $\theta_{ij}$ to describe delays in time along edge $e_i \in E$. The random variables are independent and identically distributed with Gaussian distribution $\theta_{ij} \sim N(\mu_{ij}, \sigma_{ij}^2)$, where the mean $\mu_{ij}$, $\sigma_{ij}$ are from matrix $W$.

Fig. 3 shows an example of the Gaussian time delay and the shortest-path propagation. In the worst case, the search for the shortest path takes time $O(N^2)$. Thus, the probability $P(D_{ij} | s)$ in Eq. (8) can be rewritten as follows,

$$P(D_{ij} | s) = \prod_{i=1}^{m_0} \prod_{k=1}^{m_j} \left( \prod_{\theta_{ij} \in \mathcal{G}} p(d_{ij} | \theta_{ij}) \right) = \prod_{i=1}^{m_0} P(\theta_{ij}) .$$

(9)

where $m_0$ is the number of observers during the time window, and $\theta_{ij}$ is the time delay. Based on the Gaussian distribution $\theta_{ij} \sim N(\mu_{ij}, \sigma_{ij}^2)$, the mean and variance of the random variable $\theta_{ij}$ are $\mu_{X}^{ij}$ and $\sigma_{X}^{ij}$ respectively. Let $\Lambda = (\sigma_{X}^{ij})^{-1}$, the Eq. (9) can be converted to Eq. (10),

$$P(D_{ij} | s) = \prod_{i=1}^{m_0} (2\pi \Lambda)^{-1/2} e^{-\left( t_{i} - t_{j} - \mu_{X}^{ij} / 2\Lambda \right)^2 / 2\Lambda} .$$

(10)

Similarly, for unobservers, let $\Lambda' = (\sigma_{X}^{ij})^{-1}$, we have

$$P(D_{ij} | s) = \prod_{i=1}^{m_0} (2\pi \Lambda')^{-1/2} e^{-\left( t_{i} - t_{j} - \mu_{X}^{ij} / 2\Lambda' \right)^2 / 2\Lambda'} .$$

(11)
Thus, Eq. (8) can be rewritten as
\[ P(D|s) = \prod_{i=1}^{m} \left( 2\pi A \right)^{-1/2} e^{-(t_s-t_i-\mu_s)^2/(2A)} \cdot \prod_{i=1}^{m} \left[ 1 - (2\pi A')^{-1/2} e^{-(t_s-t_i-\mu_{s'})^2/(2A')} \right]. \] (12)

Let all time delays share the same variance \( \sigma^2 \), then the log-likelihood function in Eq. (12) is
\[ \ln P(D|s) = -\frac{1}{2\lambda} \sum_{i=1}^{m} (t_s - t_i - \mu_s)^2 + \frac{1}{2\lambda} \sum_{i=1}^{m} (t_s - t_i - \mu_{s'})^2 + C, \] (13)
where \( C \) is a constant, \( t_s(t_i) - t_s^* \) is the observed (unobserved) time delay.

Thus, maximizing the log-likelihood in Eq. (13) is tantamount to minimizing a quadratic regression function in Eq. (14), where the target vector \( t \in \mathbb{R}^{m} \) is the observed time delay, i.e., \( t_s - t_i^* \) for detector \( d_k \), the coefficient matrix \( A = (\mu_1, \cdots, \mu_m) \in \mathbb{R}^{m \times m} \) is the shortest path delay w.r.t. nodes \( d_k \in D_k \). \( T \in \mathbb{R}^m \) is a vector of value \( T \), which approximates the time delay of nodes \( D_k \), i.e., \( t_s - t_i^* = (T + T^*) - T^* = T \). Matrix \( B = (\mu_1, \cdots, \mu_m) \in \mathbb{R}^{m \times m} \) denotes the shortest delay time w.r.t. inactivated nodes \( d_k \in D_k \). The parameter \( \lambda > 0 \) controls the weight. The constraints guarantee the optimal solution sparse and non-negative.

\[ s^* = \arg \min_x \frac{1}{2} \| t - A x \|^2_2 - \lambda \| T - B x \|^2_2 \quad \text{s.t.:} \quad x^T x \leq \tau, \quad x \geq 0. \] (14)

The above formulation relaxes traditional discrete optimization on graphs. Such a relaxation leads to efficient algorithms. Now we intuitively explain the above problem from the viewpoint of multi-criteria quadratic programming. The first objective function, \( \| t - A^T x \|^2_2 \), aims to fit time delay of activated detectors \( D_k \), the second objective function \( \| T - B^T x \|^2_2 \) aims to fit time delay of inactivated detectors \( D_k \). \( B \) is not actually activated during the time window \( T \), a negative parameter \( -\lambda < 0 \) is used to avoid fitting the trade-off Pareto solutions can be obtained by varying parameter \( \lambda \).

5. Online algorithm

In this section, we present an Online Stochastic Sub-gradient algorithm to solve the regression model in Eq. (14). The proposed regression model, compared to the classical regression learning, has several new challenges: 1) the dependent variable \( t \) is implicit, because the initial propagation time is unknown, 2) the \( l_1 \) non-convex objective function expects sparse and fast convergent algorithm, and data collected from detectors arrive continuously. To solve the challenges, we use the Relative Time Difference of Arrivals and Online Sub-gradient based on convex approximation as the solution.

5.1. Relative time difference

The dependent variable \( t \) in Eq. (14) is implicit, because the initial propagation time \( t^* \) is unknown. To solve this challenge, we can use an “anchor node” to cancel out the initial time \( t^* \). Assume the orth detector \( d_0 \) is the “anchor node”, its activated time is \( t_s = t^* + \sum_{i=1}^{m} c_i(t^* - d_i) \), where \( \theta_i \) is the time delay along edges \( e_i \in \mathcal{P}(s^*, d_i) \). Then, the Relative Time Difference of Arrivals (RTDA) between \( d_k \) and the \( k^{th} \) detector \( d_k \) is \( t_k^* := t_k - t^* = \sum_{i=1}^{m} c_i(t^* - d_i) \theta_i - \sum_{i=1}^{m} c_i(t^* - d_i) \theta_i \).

Based on RTDA, Eq. (14) has an elementary column change, i.e.,
\[ \tilde{A}(c_k) := A(\cdot, c_k) - A(\cdot, c_d), \quad \tilde{B}(c_k) := B(\cdot, c_k) - B(\cdot, c_d). \] (15)

Thus, the dimension of \( \tilde{A} \) and \( \tilde{B} \) are \( m - 1 \) and \( m - 1 \) respectively. Then, Eq. (14) can be relaxed to Eq. (16).

\[ s^*(x) = \arg \min_x \frac{1}{2} \| t - A^T x \|^2_2 - \lambda \| B^T x \|^2_2 + \rho \| x \|^2_1. \] (16)

5.2. Convex approximation

To address the non-convex challenge, we wish to find a sequence of convex programs, through which the non-convex function can be approximated and converge to a local minimum. To obtain a convergent sequence, at step \( k + 1 \), the concave part \(-\lambda \| B^T x \|^2_2 \) is linearly approximated by using the differential at the previous iterative point \( x_k \), i.e.,
\[ \frac{\partial}{\partial x} (-\lambda \| B^T x \|^2_2) \big|_{x_k}, \quad x \geq -2\lambda B^T B x_k. \] (17)

At step \( k + 1 \), we only solve a convex optimization as follows:
\[ x_{k+1} \leftarrow \min_{x \geq 0} \left\{ \frac{1}{2} \| t - A^T x \|^2_2 - 2\lambda B^T B x_k + \rho \| x \|^2_1 \right\}. \] (18)

Lemma 1. The non-convex program in Eq. (16) converges under iterations \( \{x_1, \cdots, x_k, \cdots\} \) generated by Eq. (18).

Proof. Denote Eq. (16) as \( f(x) \), which is a combination of the convex part \( f_{\text{conv}}(x) = \frac{1}{2} \| t - A^T x \|^2_2 \) and the concave part \( f_{\text{conc}}(x) = -\lambda \| B^T x \|^2_2 \). At step \( k + 1 \), we have \( f_{\text{conv}}(x_{k+1}) + f_{\text{conc}}(x_{k+1}) \leq f_{\text{conv}}(x_{k}) + f_{\text{conc}}(x_{k}) x_{k+1} + f_{\text{conc}}(x_{k+1}) - x_{k+1} - x_{k} \). Adding both sides of the above two inequalities, we obtain the result \( f(x_{k+1}) \leq f(x_{k}) \). Therefore, Eq. (16) under the sequence \( \{x_1, \cdots, x_k, \cdots\} \) generated by Eq. (18) is convergent. \( \square \)
5.3. Online sub-gradient

We use the sub-gradient method to solve the $l_1$ regularization problem in Eq. (16). Let $\mathcal{J}(x) = F(x) + G(x) = \frac{1}{2}\|\mathbf{F} x\|^2_2 - 2\mathcal{B}^T\mathbf{B} x$, which is an approximation of the first two parts in Eq. (16) by using the CCCP programming [25] at the concave part $-\lambda\|\mathbf{B} x\|^2_2$. The sub-gradient of $\mathcal{J}$ is as follows,

$$
\nabla_j \mathcal{J}(x) = F_j(x) + G_j(x) = \frac{1}{2} \left[ \sum_{i=1}^{m} (\bar{t}_i - a_i^T x)^2 \right]_j + \left[ -\lambda \sum_{i=1}^{m} (b_i^T x) \right]_j - \left( \mathbf{A}^T \mathbf{A} x \right)_j - 2\lambda \mathbf{B} \mathbf{B} \mathbf{x}_k.
$$

Now Eq. (16) turns to $s^*(x) = \arg\min \mathcal{J}(x) + \rho \|x\|_1^2$ and this objective function is non-differentiable. Assume $\overline{x} = (x^1, \ldots, x^s)^T$ is the optimal global point. Consider the $j$th variable $x^j$. The first-order optimality conditions are:

$$
\left\{ \begin{array}{ll}
\nabla_j \mathcal{J}(x) + \rho \mathbf{g}(\overline{x}) = 0, & \text{s.t. } \|\overline{x}\|_1 > 0 \\
\nabla_j \mathcal{J}(x) + \rho e : e \in [-1, 1], & \text{s.t. } \overline{x} = 0
\end{array} \right. \quad (19)
$$

where $\mathbf{g}(\overline{x}) = 1, \overline{x} > 0; -1, \overline{x} < 0; 0, \overline{x} = 0$.

These conditions can be used to define a sub-gradient for each $\overline{x}^j$:

$$
\nabla_s \mathcal{J}(x) = \left\{ \begin{array}{ll}
\nabla_j \mathcal{J}(x) + \rho \mathbf{g}(\overline{x}), & \|\overline{x}\|_1 > 0 \\
\nabla_j \mathcal{J}(x) + \rho \mathbf{e} : \mathbf{e} \in [-1, 1], & \|\overline{x}\|_1 = 0
\end{array} \right.
$$

Thus, based on the three solutions, the sub-gradient method uses iterations:

$$
x_{k+1} = x_k - \eta_t (\nabla_s \mathcal{J}(x)),
$$

where the parameter $\eta_t > 0$ is the learning rate. In our analysis, we only consider constant learning rate with a fixed $\eta_t > 0$.

5.4. The online stochastic sub-gradient (OSS) algorithm

In this section, we design an Online Stochastic Sub-gradient detection algorithm to continuously infer the diffusion propagations. Algorithm 1 summarizes the solution to Eq. (16), where the sparse non-convex regression is solved by iteratively calculating the convex program in Eq. (18) and the non-smooth $l_1$ program in Eq. (19). In terms of online learning, we use stochastic sub-gradient iterations to calculate $\nabla_j \mathcal{J}(x)$ during the time window $T$, i.e.,

$$
\nabla_j \mathcal{J}(x) = (\overline{t}_j - a_j x)^T (-a_j).
$$

As shown in Algorithm 1, the proposed online algorithm calls two functions for continuous learning. Function 1 corresponds to the online computation within the time window $T$ where the detectors are activated by one. Function 2 corresponds to the one-time computation after the time window $T$. The corresponding overall framework can be found in Fig. 4.

Based on the input of the proposed algorithm and the overall framework in Fig. 4, there are some assumptions in the OSS algorithm. 1) At least 1 detector needs to be activated during the time window. We used the time delay of both activated detectors (matrix $A$ and $D_0$) and inactivated detectors (matrix $B$ and $D_0$) in the proposed algorithm. During the time window, Function 1 is only called for activated detectors, and each is activated sequentially. After the time window, Function 2 is called because data from both the activated and inactivated detectors is available.

Algorithm 1 The online detection algorithm.

Input:

- $G$: Network Graph;
- $D$: Detectors $\{D_0 \cup D_0\}$;
- $u$: The propagation mean parameters $\mu = (\mu_1, \ldots, \mu_{|E|})^T$;
- $\sigma$: The propagation variance parameters $\sigma^2 = (\sigma_1^2, \ldots, \sigma_{|E|}^2)$;
- $\epsilon$: Stop criteria;

Output:

- $\mathbf{s} = \{s_1, \ldots, s_m\}$: A set of diffusion propagences $s^*$;
- $d_{\text{init}}(b) \leftarrow \text{anchor}(D_0(u))$; // randomly pick anchors
- $i = T$ then
- for each $d_i \in D_0$ do
  - $s^* \leftarrow \mathcal{O}(G, \mathcal{F}, \epsilon, \eta_t)$; // Call Function 1
  - end if
- end for
- for each $v_i \in V$ do
  - $d_{\text{init}}(b) \leftarrow \mathcal{O}(G, \mathcal{F}, \epsilon, \eta_t)$; // Call Function 2
  - end if
- end for
- end for
- end if
- return $s^* \leftarrow \mathcal{O}(G, \mathcal{F}, \epsilon, \eta_t)$;

Function 1 Online detecting during the time window.

Input:

- $A$: Observation matrix;
- $T$: Time delay;
- $\epsilon$: Stop criteria;
- $\eta_t$: Learning rate;

Output:

- $\mathbf{x}^* = \{x_1, \ldots, x_j\}$: The indicator vector $x^*$;
- $n_{t+1} \leftarrow$ a best guess;
- $\mathcal{L}(x) = \frac{1}{2}\|\mathbf{F} - \mathbf{A} x\|^2_2$
- $\mathbf{s}^*(x) \leftarrow \arg\min \mathcal{L}(x) + \rho \|x\|_1^2$ $x_0$;
- repeat
- $n_{t+1} \leftarrow n_{t+1} - \eta_t (\nabla_s \mathcal{J}(x))$;
- end for
- $\|n_{t+1} - x_0\|_2 \leq \epsilon$
- $\mathbf{x}^* \leftarrow \sum_{t=1}^{T_1} \mathbf{x}_t$ in a descending order;
- $\mathbf{x}^* \leftarrow \mathbf{x}_t$ in a descending order;
- $\mathbf{x}^* \leftarrow \mathbf{x}_t$ in a descending order;
- $\mathbf{x}^* \leftarrow \mathbf{x}_t$ in a descending order;

the proposed algorithm can be used to infer diffusion propagences when there is at least one active detector (i.e., the activated detector set $D_0$ is non-null and Function 1 can be called). The network structure is known prior, i.e., the network graph $G$, the propagation mean is $\mu$ and variance is $\sigma$.
Fig. 4. An illustration of the OSS algorithm. Detectors are split into two sets. The first set is observed (activated) within the time window, and the second set is unobserved (inactivated) outside the time window. Function 1 is called by OSS within the time window $T$, and Function 2 is called after the time window $T$.

**Function 2** Online detecting after the time window.

**Input:**

- $\mathbf{A}$: Observation matrix;
- $\mathbf{B}$: Unobservation matrix;
- $\mathbf{T}$: Time delay;
- $\epsilon$: Stop criteria;
- $\eta_t$: Learning rate;

**Output:**

$$x^* = \{x_1, \ldots, x_j\}$$

1. $x_{k+1} \leftarrow$ a best guess;
2. repeat
3. $x_k \leftarrow x_{k+1}$;
4. $x_{k+1} \leftarrow \{x_k - \eta_t (\mathbf{w}^*(x))\}$;
5. until $\|x_{k+1} - x_k\| \leq \epsilon$
6. $x^* \leftarrow x_{k+1}$ in a descending order;
7. $j^* = \arg\max_j |x_j - x_{j-1}|$;
8. return $x^* = \{x_1, \ldots, x_{j^*}\}$

**Function 1** During the time window $T$, detectors are activated sequentially. To conduct continuous detection, once a detector is activated, the regression learning model is used for provenance estimation. At this stage, data from the inactivated nodes is unavailable, the parameter $\lambda$ in Eq. (16) equals 0, i.e.,

$$s^*(x) = \arg\min_{x \in \mathbb{R}^E} \frac{1}{2} \| \mathbf{r} - \mathbf{A}^T x \|_2^2 + \rho \| x \|_1.$$  \hspace{1cm} (23)

Each row of $\mathbf{A}$ and $\mathbf{r}$ in Eq. (23) denotes the shortest path and time delay from an observed detector to other detectors. By choosing an anchor node, Eq. (15) is used to calculate the matrix $\mathbf{A}$ and $\mathbf{r}$ in Eq. (23). Due to the increasing of the number of activated detectors, matrix $\mathbf{A}$ and $\mathbf{r}$ in Eq. (23) increase dynamically.

**Function 2** After the time window $T$, data from both the activated and inactivated nodes is available. Thus, we can use Eq. (16) to locate the diffusion provenances.

We use Fig. 4 to show the procedure of the online learning: (1) During the time window $T$, the detectors are activated one by one. Once a detector is activated, we use Function 1 to estimate the provenance; (2) After the time window, we obtain all the activated detectors and inactivated detectors, and Function 2 can be used for detection.

**Example:** Consider a network in Fig. 5. Assume $t^* = 1, D = 3$, and the time window $T$ is $[1, 5]$. Also, we assume detectors $S_1, S_2$ and $S_3$ are observed at time $t_1 = 1, t_2 = 3$ and $t_3 = 5$ respectively. Let $S_1$ be the anchor node. At $t = 1$, we have $A = \begin{bmatrix} 1 & 0 & 1 & 2 & 4 & 4 & 4 \\
1 & 2 & 2 & 0 & 2 & 2 & 4 \end{bmatrix}^T$. Then, at $t = 3$, $S_2$ is activated, and we update the matrix $A$ and $t$ as follows,

$$A = \begin{bmatrix} 1 & 0 & 1 & 2 & 4 & 4 & 4 \\
1 & 2 & 2 & 0 & 2 & 2 & 4 \end{bmatrix}^T, \quad t = \begin{bmatrix} 1 - t^* \\
3 - t^* \end{bmatrix}.$$

The $A$ and $t$ are used to calculate $\mathbf{A}$ and $\mathbf{r}$ by Eq. (12),

$$\mathbf{A} = \begin{bmatrix} 0 & 2 & 1 & -2 & -2 & 0 & -2 \\
2 & 4 & 3 & 0 & -2 & 4 & 2 \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} 2 \\
4 \end{bmatrix}.$$

At the last step, $t = 5$ and $S_3$ is observed. The variables $\mathbf{A}$ and $\mathbf{r}$ are continuously updated to

$$\mathbf{A} = \begin{bmatrix} 0 & 2 & 1 & -2 & -2 & 0 & -2 \\
2 & 4 & 3 & 0 & -2 & 4 & 2 \end{bmatrix}^T, \quad \mathbf{r} = \begin{bmatrix} 2 \\
4 \end{bmatrix}.$$
The corresponding online solutions of x approximate the offline optimal solution when the time window T ends. We will leave the competitive boundary analysis in the future work. We have used Fig. 5 to show how to calculate matrix $\mathbf{A}$, $\mathbf{B}$ and $t$ in the above example. Fig. 5 shows the shortest path from nodes $\{S_0, \cdots, S_7\}$ to anchor node $S_1$. For example, the elements value in matrix $\mathbf{A}$ are based on calculating shortest path to $S_1$ in Fig. 5.

### 6. Experiments

In this section, we report experimental results on two synthetic network datasets and two real-world datasets. The experiments are designed to validate (1) the optimal parameters for the new model, (2) the superiority of the proposed model compared with benchmark methods, and (3) the performance of the proposed methods.

#### 6.1. Experimental data

We use two synthetic datasets (Albert-Barabasi [26] and Small World [27]) and two real-world datasets (Twitter and Facebook from SNAP \(^3\)) for parameter study, performance testing and algorithm comparison. The parameter settings are listed in Table 1.

Albert-Barabasi generates random scale-free networks using a preferential attachment mechanism. The network begins with an initially connected network containing $\beta_0$ nodes. New nodes are added to the network one at a time. Each new node is connected to $\beta \leq \beta_0$ existing nodes with a probability proportional to the number of links that existing nodes already have. Formally, the probability $\pi_i$ that the new node is connected to node $i$ is $\pi_i = k_i / \sum k_j$, where $k_i$ is the degree of node $i$ and the sum is made over all pre-existing nodes $j$. In this model, we set the parameter $n = 4$, which denotes the number of edges created by each new node.

Small World is defined as a network in which the typical distance $\xi$ between two randomly chosen nodes (the number of steps required) grows proportionally to the logarithm of the number of nodes $N$ in the network, that is $\xi \sim \log N$. We use parameter $\alpha$ to denote that each node is connected to $\alpha$ nearest neighbors in the topology, and $p$ denotes the rewiring probability. In particular, we set $\alpha = 4$ and $p = 0.1$ in our experiments.

The original datasets only contain network structure information without time propagation labels. We use Gaussian distribution $N(1, 0.001)$ as the time delay distribution. We simulated the propagation by randomly setting five diffusion sources.

We used the Independent Cascade Model to generate cascades. When node $u$ becomes active, it has a single chance of activating each currently inactive neighbor $v$. The attempt to activate succeeds independently with a probability of $p_{uv}$, set to $p_{uv} = 0.5$ as a constant in our experimental setting. Because the cascades are generated by the Independent Cascade Model, each node has a single chance of activating each currently inactive neighbor, so this is one-to-one communication. Thus, the datasets can be considered as half-real data.

#### 6.2. Experimental setup

We assess our methods w.r.t. the average distance (hops) between actual locations of the diffusion provenances and the estimated locations. Smaller hops indicate that the algorithms have higher accuracy. Let $S^*$ denote the actual provenance, and the estimated provenance set as $\hat{S}$. Since the size of $S^*$ may not be equal to that of $\hat{S}$, we measure their distance $h(S^*, \hat{S})$ by calculating the average number of hops between each element $s_i \in S^*$ and its closest neighbor $\hat{s}_i \in \hat{S}$, $h(S^*, \hat{S}) = \frac{1}{|S^*|} \sum_{i=1}^{\frac{|S^*|}{2}} ||s_i - \hat{f}(S^*, \hat{S})||^2$, where $\hat{f}(S^*, \hat{S})$ selects the node in $\hat{S}$ that is closest to $s_i$, and $||s_i - \hat{f}(S^*, \hat{S})||^2$ denotes the hops between nodes $s_i$ and $\hat{f}(S^*, \hat{S})$.

#### 6.3. Experimental results

**Parameter study.** We first test the parameters in Eq. (16) w.r.t. the number of diffusion provenances $k$, the number of detectors $m$, the length of monitoring time window $T$, and the parameters $\lambda$ and $\rho$. The default parameters are: the number of diffusion provenances $k = 5$, the number of detectors $d = 20\% \times N$, the monitoring time window $T = 5$ minutes. The selection of provenances and detectors are random. The propagation time delay along each edge follows a Gaussian $\theta_i \sim N(1, 0.001)$.

#### 6.3.1.** Experimental results

**Parameter study.** We first test the parameters in Eq. (16) w.r.t. the number of diffusion provenances $k$, the number of detectors $m$, the length of monitoring time window $T$, and the parameters $\lambda$ and $\rho$. The default parameters are: the number of diffusion provenances $k = 5$, the number of detectors $d = 20\% \times N$, the monitoring time window $T = 5$ minutes. The selection of provenances and detectors are random. The propagation time delay along each edge follows a Gaussian $\theta_i \sim N(1, 0.001)$. Fig. 6 demonstrates the model performance w.r.t. different parameters on the synthetic and real-world datasets. We replicated the algorithms five times and used the mean as the evaluation indicator.

The number of diffusion provenances $k$. From Fig. 6(a), we can see that the error rate (hops) decreases w.r.t. the number of diffusion provenances. We can conclude that the more diffusion provenances, the higher probability that at least one provenance will be found.

The number of detectors $m$. Fig. 6(b) shows the average distance (hops) w.r.t. the number of detectors. The result demonstrates that the accuracy improves w.r.t. the number of detectors.

The monitoring time window $T$. The monitoring time window indicates the detection time span. Fig. 6(c) shows that the error hops drop with the time window.

The parameter $\lambda$. The parameter $\lambda$ controls the preference between the activated nodes (convex part) and the inactivated nodes (concave part). As shown in Fig. 6(d), the parameter $\lambda$ should weigh the convex part and the concave part. If $\lambda$ is selected too large, it overfits the inactivated nodes; otherwise, it overfits the activated nodes.

The parameter $\rho$. $\rho > 0$ is the regularization parameter, and restricts the size of $x$. If $\rho$ is too large, the algorithm incurs time costs, especially on large networks. From Fig. 6(e), we observe the minimal hops when $\rho$ is 6.

Continuous detection. We test the proposed online algorithm on both the synthetic and real-world datasets. Also, we empirically study the learning rate parameter $\eta_t$ in the sub-gradient algorithm.

### Table 1

List of the four datasets.

<table>
<thead>
<tr>
<th>DataSet</th>
<th>Nodes</th>
<th>Edges</th>
<th>Avg. degree</th>
<th>Avg. path length</th>
<th>Avg. clustering coefficient</th>
<th>Other parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albert-Barabasi</td>
<td>1000</td>
<td>3990</td>
<td>7.98</td>
<td>3.172</td>
<td>0.037</td>
<td>$n = 4$</td>
</tr>
<tr>
<td>Small World</td>
<td>1000</td>
<td>3999</td>
<td>7.99</td>
<td>5.079</td>
<td>0.473</td>
<td>$\alpha = 4, \rho = 0.1$</td>
</tr>
<tr>
<td>Twitter</td>
<td>10,269</td>
<td>36,173</td>
<td>30.54</td>
<td>5.702</td>
<td>0.627</td>
<td>$\mu = 0.01, \sigma = 0.01$</td>
</tr>
<tr>
<td>Facebook</td>
<td>35,020</td>
<td>30,527</td>
<td>19.78</td>
<td>3.892</td>
<td>0.671</td>
<td>$\mu = 0.01, \sigma = 0.01$</td>
</tr>
</tbody>
</table>

\[ \mathbf{B} = \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & -2 & 0 \\ 2 & 4 & 3 & 0 & -3 & -2 & 2 & -4 \end{bmatrix}^T. \]

\[ h(S^*, \hat{S}) = \frac{1}{|S^*|} \sum_{i=1}^{\frac{|S^*|}{2}} ||s_i - \hat{f}(S^*, \hat{S})||^2. \]
Fig. 7 demonstrates the OSS algorithm with learning rates $\eta_i$. The algorithm reaches the least average distance when $\eta_i = 0.1$.

**Comparison with benchmark methods.** We compare the proposed non-convex sparse regression model (NSR) with three benchmark methods: 1) the convex-based sparse regression method (VEXR) [28], which uses activated nodes for regression and locating; 2) the concave-based sparse regression method (CAVR) [28], which uses inactivated nodes for regression; 3) Betweenness centrality [21], which measures a node's centrality in a network. We use betweenness centrality to measure how often a node appears on the shortest path. The betweenness centrality of a node $v$ is given by the function: $g(v) = \sum_{s \neq t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where $\sigma_{st}$ is the total number of the shortest paths from node $s$ to node $t$ and $\sigma_{st}(v)$ is the number of those paths that pass through $v$; and 4) Bonacich centrality [23], also known as Eigenvector Centrality, measures the influence of a node in a network. The influence of nodes is based on assigning relative scores to all nodes in the network. The centrality score of node $v$ is defined as: $x_v = \frac{1}{2} \sum_{i=1}^{N} A_{vi} x_i = \frac{1}{2} \sum_{i=1}^{N} A_{vi} x_i$, where $G$ is network, $A = (a_{ij})$ be the adjacency matrix, i.e., $a_{ij} = 1$ if node $v$ is linked to node $t$, and $a_{ij} = 0$ otherwise, $M(v)$ is a set of the neighbors of $v$, and $\lambda$ is a constant (we set it as 1 in our experiment). We use UCINET 6 for Windows to compute the Bonacich centrality.

We compare the four methods on the two synthetic networks and two real-world networks. The parameters are set as default. We test two settings, the number of diffusion provenances $k = 5$ and $k = 10$ on the four synthetic datasets. For each group, we randomly sample different numbers of detectors $m$. From the results in Table 2, the results can be summarised as follows.

1. The proposed NSR model performs better than both CVXR and CAVR models on most of the datasets, by leveraging both activated and inactivated nodes and achieves the low-

![Fig. 6. Parameter study on synthetic and real-world datasets by using the proposed regression learning model and Online Stochastic Sub-gradient algorithm. The number of hops (error rate) w.r.t. (a) the diffusion provenances number $k$; (b) the detector number $m$; (c) the monitoring time window $T$; (d) the parameter $\lambda$; and (e) the parameter $\rho$. The average distance on synthetic/real-world datasets w.r.t propagation time.](image-url)
Fig. 7. Parameter $\eta_i$ in the proposed Online Stochastic Sub-gradient OSS algorithm.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Parameter $\eta_i$</th>
<th>$k = 5$</th>
<th>$k = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>0.05</td>
</tr>
<tr>
<td>Albert-Barabasi</td>
<td>NSR</td>
<td>3.47±0.64</td>
<td>2.33±0.66</td>
</tr>
<tr>
<td></td>
<td>CVXR</td>
<td>3.48±0.25</td>
<td>2.37±0.31</td>
</tr>
<tr>
<td></td>
<td>Betweenness</td>
<td>3.82±0.66</td>
<td>2.77±0.68</td>
</tr>
<tr>
<td></td>
<td>Bonacich</td>
<td>3.48±0.34</td>
<td>3.19±0.26</td>
</tr>
<tr>
<td>Small-World</td>
<td>NSR</td>
<td>3.82±0.31</td>
<td>3.30±0.33</td>
</tr>
<tr>
<td></td>
<td>CVXR</td>
<td>4.31±0.56</td>
<td>3.32±0.68</td>
</tr>
<tr>
<td></td>
<td>Betweenness</td>
<td>4.44±0.48</td>
<td>3.51±0.20</td>
</tr>
<tr>
<td></td>
<td>Bonacich</td>
<td>4.53±0.41</td>
<td>3.62±0.27</td>
</tr>
<tr>
<td>Twitter</td>
<td>NSR</td>
<td>3.27±0.38</td>
<td>3.46±0.48</td>
</tr>
<tr>
<td></td>
<td>CVXR</td>
<td>4.33±0.51</td>
<td>3.67±0.24</td>
</tr>
<tr>
<td></td>
<td>Betweenness</td>
<td>4.49±0.43</td>
<td>4.39±0.60</td>
</tr>
<tr>
<td></td>
<td>Bonacich</td>
<td>3.30±0.50</td>
<td>3.45±0.44</td>
</tr>
<tr>
<td>Facebook</td>
<td>NSR</td>
<td>3.66±0.45</td>
<td>3.66±0.55</td>
</tr>
<tr>
<td></td>
<td>CVXR</td>
<td>3.77±0.69</td>
<td>4.25±0.65</td>
</tr>
<tr>
<td></td>
<td>Betweenness</td>
<td>4.60±0.24</td>
<td>3.75±0.27</td>
</tr>
<tr>
<td></td>
<td>Bonacich</td>
<td>3.5±0.31</td>
<td>3.20±0.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>Albert-Barabasi</td>
<td>NSR</td>
<td>3.65±0.27</td>
<td>3.67±0.35</td>
</tr>
<tr>
<td></td>
<td>CVXR</td>
<td>3.77±0.43</td>
<td>3.67±0.35</td>
</tr>
<tr>
<td></td>
<td>Betweenness</td>
<td>3.37±0.59</td>
<td>3.21±0.23</td>
</tr>
<tr>
<td></td>
<td>Bonacich</td>
<td>3.50±0.33</td>
<td>3.33±0.25</td>
</tr>
</tbody>
</table>

Table 2
Comparisons on the four datasets.
the diffusion’s source. The accuracy increase is because the more provenances there are, the greater the probability that the influential nodes are provenances.

(6) The NSR model performs better than other baselines especially when $m$ is large. This is because that NSR carries out estimating provenances with more detectors (activated and inactivated), where more information for the regression model can be provided.

Comparison to the online algorithm with different propagation processes. We evaluate our algorithm using various propagation processes, i.e., Independent Cascades propagation shown in Fig. 8(a) and Linear Threshold propagation shown in Fig. 8(b). For the former, the corresponding description can be found in Section 6.1. For the latter, a node $v$ has threshold $\theta_v \sim U[0, 1]$, and is influenced by each neighbor $t$ according to a weight $b_{v,t}$ such that: $\sum_{t} \text{neighbor of } v b_{v,t} \leq 1$. A node $v$ becomes active when at least $\theta_v$ fraction of its neighbors are active $\sum_{t} \text{active neighbor of } v b_{v,t} \geq \theta_v$.

Fig. 8 shows the performance of the algorithm OSS with different propagation process. The hops reduce along with the propagation time because more activated detectors lead to better performance. At the end of the time window, the hops shrink as the inactivated detectors are obtained.

Online algorithm time cost. Fig. 9 plots the average running time of the algorithm OSS on the synthetic and real-world datasets. The running time raises moderately at the beginning due to the increase of activated nodes (Function 1), and then increases sharply in the end because all the activated and inactivated detectors are used for calculation (Function 2).

7. Conclusions

This paper discusses a solution for discovering the diffusion provenances in social networks in an online setting. We proposed a real-time source detection algorithm by placing detectors randomly across a network. Our approach converts the problem to a regression problem, and uses an online stochastic sub-gradient algorithm to solve regression as the information is gathered from detectors in real time. This work focuses on online detection rather than offline, to meet the practical need for early warning, real-time awareness and real-time responses to malicious information spreading within an online social network. This work can therefore be used in time-critical security monitoring applications, such as locating false rumors in business areas.

Although our algorithm for detecting the diffusion provenances detection achieves high accuracy and meets the need for a real-time response, some limitations exist that need improvement in future work: 1) the simulation study used real network data but the propagation process was synthetic, dubbed half-real data, so experiments on real network propagation data need to be undertaken; and 2) the proposed algorithm, based on sub-gradients, is a straightforward solution for online learning problems, however more state-of-the-art online learning algorithms could be applied to this online diffusion provenance problem, such as passive-aggressive (PA), second-order perceptron (SOP) and confidence-weight learning (CW).

This work inspires some interesting directions for future research: 1) the problem could be further extended by using popular stochastic epidemic models such as the SR and SRI models; and 2) previous mobile social network mining techniques could be used to harness geographical information to identify a culprit’s physical location.

Acknowledgments

This work was supported in part by the China Scholarship Council Foundation (No. 201206410056), the National Natural Science Foundation of China (NSFC) under Grant No. 61301185, No. 61370070 and No. 61671048 and the Australian Research Council (ARC) Discovery Projects under Grant No. DP140100545 and No. DP140102206.
References


Haishuai Wang received his Master’s degree in computer science from Beijing Jiaotong University, Beijing, China, in 2013. Since August 2013, he has been working toward a PhD degree in the Centre for Quantum Computation and Intelligent Systems (QCIS), University of Technology Sydney, Australia. His research focuses on data mining and machine learning.

Jia Wu received the PhD degree in computer science from the University of Technology Sydney (UTS), Australia. He is currently a Research Associate in the Centre of Quantum Computation and Intelligent Systems (QCIS), UTS. His research focuses on data mining and machine learning. Since 2009, Dr. Wu has published more than 50 refereed journals (such as IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Cybernetics, ACM Transactions on Knowledge Discovery Data, Pattern Recognition and conference papers (such as International Joint Conference on Artificial Intelligence (IJCAI), AAAI Conference on Artificial Intelligence (AAAI), IEEE International Conference on Data Mining (ICDM), IEEE International Conference on Data Engineering (ICDE), SIAM International Conference on Data Mining (SDM), ACM Conference on Information and Knowledge Management (CIKM)) in these areas, including one ICDM Best Paper Candidate Award. He is a member of the IEEE.

Shirui Pan received the PhDs degree in computer science from the University of Technology Sydney (UTS), Sydney, NSW, Australia, in 2015. He is a Research Associate with the Centre for Quantum Computation and Intelligent Systems, UTS. He has published over 20 research papers in top-tier journals and conferences, including the IEEE Transactions on Knowledge and Data Engineering, the IEEE Transactions on Cybernetics, Pattern Recognition, International Joint Conference on Artificial Intelligence, International Conference on Data Engineering, International Conference on Data Mining, SIAM International Conference on Data Mining, International Conference on Information and Knowledge Management, and Pacific-Asia Conference on Knowledge Discovery and Data Mining. His current research interests include data mining and machine learning.

Peng Zhang received his PhD degree in July 2009 from the Chinese Academy of Sciences. Since then, Dr. Zhang has worked with two universities in the USA and one national laboratory at the Chinese Academy of Sciences. He won the Best Paper Award at ICCS-14 held in Queensland, Australia. In January 2014, he joined the QCIS research center, University of Technology Sydney (UTS) as a Lecturer. Dr. Zhang is an Associate Editor in two Springer journals, Journal of Big Data, and Annals of Data Science. To date, Dr. Zhang has published over 100 papers, including PR, IEEE TKDE, KDD, ICDM, SDM, AAAI, CIKM, WWW, PAKDD, etc. He has been serving as a PC member (reviewer) in IEEE TKDE, ACM TKDD, KDD, IJCAI, ICDM etc.

Ling Chen received her Ph.D. in 2008, in Computer Engineering, from Nanyang Technological University, Singapore. She is currently a Senior Lecturer with Faculty of Engineering and Information Technology, University of Technology Sydney. Prior to that, she was a Postdoc Research Fellow with L3S Research Center, Leibniz University Hannover, Germany. Ling’s main research interests include data mining and analytics, machine learning and artificial intelligence.