Learning Influence Probabilities In Social Networks

Amit Goyal
University of British Columbia
Vancouver, BC, Canada
goyal@cs.ubc.ca

Francesco Bonchi
Yahoo! Research
Barcelona, Spain
bonchi@yahoo-inc.com

Laks V. S. Lakshmanan
University of British Columbia
Vancouver, BC, Canada
laks@cs.ubc.ca

ABSTRACT

Recently, there has been tremendous interest in the phenomenon of influence propagation in social networks. The studies in this area assume they have as input to their problems a social graph with edges labeled with probabilities of influence between users. However, the question of where these probabilities come from or how they can be computed from real social network data has been largely ignored until now. Thus it is interesting to ask whether from a social graph and a log of actions by its users, one can build models of influence. This is the main problem attacked in this paper. In addition to proposing models and algorithms for learning the model parameters and for testing the learned models to make predictions, we also develop techniques for predicting the time by which a user may be expected to perform an action. We validate our ideas and techniques using the Flickr data set consisting of a social graph with 1.3M nodes, 40M edges, and an action log consisting of 35M tuples referring to 300K distinct actions. Beyond showing that there is genuine influence happening in a real social network, we show that our techniques have excellent prediction performance.

Categories and Subject Descriptors H.2.8 [Database Management]: Database Applications - *Data Mining*
General Terms: Algorithms
Keywords: Social networks, Influence, Viral marketing.

1. INTRODUCTION

In recent years, there has been tremendous interest in the phenomenon of influence exerted by users of an online social network on other users and in how it propagates in the network. The idea is that when a user sees their social contacts performing an action such as joining an online community (say TapIt[1]), that user may decide to perform the action themselves. In truth, when a user performs an action, she may have any one of a number of reasons for doing so: she may have heard of it outside of the online social network and may have decided it is worthwhile; the action is very popular (e.g., buying an iPhone 4G may be such an action); or she may be genuinely influenced by seeing her social contacts perform that action. If there is genuine influence, it can be leveraged for a number of applications, arguably the most famous among which is viral marketing [4, 13, 9]. Other applications include personalized recommendations [17, 16] and feed ranking in social networks [15]. Besides, patterns of influence can be taken as a sign of user trust and exploited for computing trust propagation [6, 23, 5, 18] in large networks and in P2P systems.

While many of the applications mentioned above essentially assume that influence exists as a real phenomenon, two key pieces are missing from the picture. First, while some empirical studies have reported evidences of influence propagating on the social linkage [2, 7], others authors have challenged the fact that influence really exists and that it propagates between users. Indeed, Watts [22, 20, 21] challenges the very notion of influential users that are often assumed in viral marketing papers. As well, in a recent paper, Anagnostopoulos *et al.* [1] have developed techniques for showing that influence is *not* genuine and using them, showed that in the Flickr tagging data, while there is substantial social correlation in tagging behavior, it cannot be attributed to influence. This raises the question, is there evidence of genuine influence in any real social network data? The second missing piece is a systematic study of models of influence. In particular, all viral marketing papers assume that they are given as input a social graph with edges labeled by the probability with which a user's action will be influenced by her neighbor's actions. *To our knowledge, the question how or from where one can compute these probabilities of influence has been largely left open.*

In this paper, our goal is to address both the issues above: we devise various probabilistic models of influence, and w.r.t. them we show that influence is genuinely happening in a real-world social network.

The starting observation is that while real social networks don't come with edges labeled with influence probabilities, they do come with an *action log*. Informally, an action log is a table that chronicles any actions performed by every user. It is thus interesting to ask whether by analyzing the action log together with the network, we can study the two questions above. In undertaking such a study, two things should be kept in mind. First, any models proposed for influence should be compatible with the assumptions made in

[1]TapIt is a water bottle refilling network founded in 2008 to give people free access to clean sustainable water on the go: `tapitwater.com`.

WSDM'10, February 4–6, 2010, New York City, New York, USA.

applications such a viral marketing. Viral marketing papers typically assume a diffusion model of influence which satisfies a property known as submodularity. While we defer a formal definition to Section 4, intuitively it can be thought of as a law of diminishing returns. Second, the action log is huge in size. Thus, any algorithms developed for learning and testing the influence models should make minimal number of scans over this data.

In this paper, we make the following contributions:

- We propose a solution framework. By starting with the diffusion models assumed in viral marketing, we elicit desiderata for models of probabilistic influence. These include a mandatory *submodularity* property and a desirable, but not mandatory *incrementality* property.
- We propose a variety of probabilistic models of influence between users in a social network. We show that all of them satisfy submodularity while all with the exception of one satisfies incrementality. Intuitively, an incremental model allows efficient testing.
- We develop algorithms for learning (the parameters of) all the proposed models, taking as input a social graph and an action log. We optimize the scans and show that our algorithms can learn all the models in no more than two scans. A highlight is that some of our models let us predict not just whether a user will perform an action but the time by which she will perform it.
- One of the most accurate models is what we call *continuous time* model. Unfortunately, it is the most expensive to test (it's not incremental). To mitigate this, we develop an approximate model called *discrete time* model, which is incremental and is much more efficient to test.
- It turns out evaluating (testing) the learned models is far from trivial. Thereto, we develop algorithms for testing all the models learned. The algorithms require just one scan over the action log.
- Last but not the least, we put the models and algorithms to test on the Flickr data set where for actions, we take users joining online communities. The data set consists of a graph with 1.3M nodes and more than 40M edges and an action log with 35M tuples referring to 300K different actions. Our results show there is genuine influence between users. In particular, we introduce the metrics of *user influenceability* and *action influence quotient*. Users (actions) with high values for this metric do experience genuine influence compared to those with low values. Our results also show that all proposed models have a reasonable performance, and continuous time model has the best performance. Discrete time model has an almost identical performance while it is much more efficient to test. We also show that our models can predict the time at which a user will perform an action with an impressive error margin.

Section 2 provides relevant background and discusses related work. In Section 3, we give a formal statement of the problem studied while in Section 4, we develop a solution framework and in Section 5 we present the models for probabilistic influence. Section 6 presents the algorithms for learning the models and for evaluating them, while Section 7 presents results from an extensive set of experiments.

2. BACKGROUND AND RELATED WORK

Suppose we are given a social network together with the estimates of reciprocal influence between individuals in the network, and suppose that we want to push a new product in the market. The idea behind *viral marketing* is that by targeting the most influential users in the network we can activate a chain-reaction of influence driven by word-of-mouth, in such a way that with a very small marketing cost we can actually reach a very large portion of the network. The mining problem of *influence maximization* is the following: given such a network with influence estimates, how to select an initial set of k users such that they eventually influence the largest number of users in the social network.

Domingos and Richardson [4, 13] were the first to consider the propagation of influence and the problem of identification of influential users by a data mining perspective. The problem is tackled by means of a probabilistic model of interaction, and heuristics are given for choosing the users.

Kempe *et al.* [9] attacked roughly the same problem as a problem in discrete optimization (which is known to be NP-complete), obtaining provable approximation guarantees in several preexisting models coming from mathematical sociology. In particular their work focuses on two fundamental propagation models, namely *Linear Threshold Model* and *Independent Cascade Model*. They also proposed a broader framework that that simultaneously generalizes the Linear Threshold and Independent Cascade models, and that has equivalent formulations in terms of thresholds and cascades. We next recall the threshold formulation.

General Threshold Model. At a given timestamp, each node is either active (an adopter of the innovation, or a customer which already purchased the product) or inactive, and each node's tendency to become active increases monotonically as more of its neighbors become active. Time unfolds deterministically in discrete steps. As time unfolds, more and more of neighbors of an inactive node u may become active, eventually making u become active, and u's activation may in turn trigger further activations by nodes to which u is connected. In the General Threshold Model each node u has a monotone activation function $f_u : 2^{N(u)} \rightarrow [0, 1]$, from the set of neighbors N of u, to real numbers in $[0, 1]$, and a threshold θ_u, chosen independently and uniformly at random from the interval [0,1]. A node u becomes active at time $t + 1$ if $f_u(S) \geq \theta_u$, where S is the set of neighbors of u that are active at time t.

Under all the propagation models discussed in [9], the influence maximization problem is shown to be NP-hard. Kempe *et al.* however show that the *influence spread* of a set of nodes is a function with the nice features of being monotone and submodular (see Section 4). Exploiting these properties they present a greedy approximation algorithm. Indeed, for any monotone and submodular function f with $f(\emptyset) = 0$, the problem of finding a set S of fixed cardinality k such that $f(S)$ is maximal, can be approximated by a greedy algorithm within a factor of $(1 - 1/e)$, as shown in [11].

A limitation of [9] is the efficiency of their greedy algorithm, which requires to compute the influence spread given a seed set. For this difficult task they run Monte-Carlo simulations of the propagation model for sufficiently many times

to obtain an accurate estimate, resulting in very long computation time. A recent line of research [10, 3] has started developing methods for improving the efficiency of the greedy algorithm for influence maximization.

Leskovec *et al.* [10] study the propagation problem from a different perspective namely *outbreak detection*: how to select nodes in a network in order to detect the spread of a virus as fast as possible? They present a general methodology for near optimal sensor placement in these and related problems. By exploiting submodularity they develop an efficient algorithm based on a "lazy-forward" optimization in selecting new seeds, achieving near optimal placements, while being 700 times faster than the simple greedy algorithm. In spite of this big improvement over the basic greedy algorithm, their method still faces serious scalability problems as shown in [3]. In that paper, Chen *et al.* improve the efficiency of the greedy algorithm and propose new degree discount heuristics that produce influence spread close to that of the greedy algorithm but much more efficiently.

All the papers discussed above assume the basic framework and propagation models of [9], where the influence probabilities $p_{v,u}$ on the edges are given as input. In this paper instead we study how this probabilities can be produced by mining past influence cascades, or in other terms, the past behavior of users.

Tang *et al.* [19] introduce the problem of topic-based social influence analysis. Given a social network and a topic distribution for each user, the problem is to find topic-specific subnetworks, and topic-specific influence weights between members of the subnetworks. They propose a Topical Affinity Propagation (TAP) approach using a graphical probabilistic model. They also deal with the efficiency problem by devising a distributed learning algorithm under the Mapreduce programming model. Moreover, they also discuss the applicability of their approach to the *expert finding* problem.

Independently and concurrently with us, Saito *et al.* [14][2] have studied the same problem we tackle in this paper, focussing on the Independent Cascade model of propagation. They formally define the likelihood maximization problem and then apply EM algorithm to solve it. While their formulation is elegant, it is not scalable to huge datasets like the one we are dealing in this work. This is due to the fact that in each iteration, the EM algorithm must update the influence probability associated to each edge.

3. PROBLEM DEFINITION

We are given a *social graph* in the form of an undirected graph $G = (V, E, \mathcal{T})$ where the nodes V are users. An undirected edge $(u, v) \in E$ between users u and v represents a social tie between the users. $\mathcal{T} : E \rightarrow \mathbb{N}$ is a function labeling each edge with the timestamp at which the social tie was created.[3] We're also given an *action log*, a relation $Actions(User, Action, Time)$, which contains a tuple (u, a, t_u) indicating that user u performed action a at time t_u. It contains such a tuple for every action performed by every user of the system. We will assume that the projection of *Actions* on the first column is contained in the set of nodes V of the social graph G. In other words, users in the *Actions* table correspond to nodes of the graph. We let $\mathcal{A}$ denote the universe of actions. In the following, we assume for ease of exposition that a user performs an action at most once. We denote with A_u the number of actions performed by user u in the training set, with $A_{u\&v}$ the number of actions performed by both u and v in the training set, with $A_{u|v}$ the number of actions either u or v performs in the training set. Clearly, $A_{u|v} = A_u + A_v - A_{u\&v}$. We also use A_{v2u} to denote the number of actions propagated from v to u in the training set. We next define propagation of actions.

Definition 1 (Action propagation). *We say that an action $a \in \mathcal{A}$ propagates from user v_i to v_j iff: (i) $(v_i, v_j) \in E$; (ii) $\exists (v_i, a, t_i), (v_j, a, t_j) \in Actions$ with $t_i < t_j$; and (iii) $\mathcal{T}(v_i, v_j) \leq t_i$. When this happens we write $prop(a, v_i, v_j, \Delta t)$ where $\Delta t = t_j - t_i$.*

Notice that there must be a social tie between v_i and v_j, both must have performed the action after the moment in which their social tie was created. This leads to a natural notion of a propagation graph, defined next.

Definition 2 (Propagation graph). *For each action a, we define a propagation graph $PG(a) = (V(a), E(a))$, as follows. $V(a) = \{v \mid \exists t : (v, a, t) \in Actions\}$; there is a directed edge $v_i \xrightarrow{\Delta t} v_j$ in $E(a)$ whenever $prop(a, v_i, v_j, \Delta t)$.*

The propagation graph consists of users who performed the action, with edges connecting them in the direction of propagation. Observe that the propagation graph is a DAG. Each node can have more than one parent; it is directed, and cycles are impossible due to the time constraint which is the basis for the definition of propagation. Note that the propagation graph can possibly have disconnected components. In other words, the propagation of an action is just a directed instance (a flow) of the undirected graph G, and the log of actions $Actions(User, Action, Time)$ can be seen as a collection of propagations. When a user performs an action, we say that it is activated w.r.t. that action. Once a user activates, it becomes contagious and cannot de-activate. It may now influence all its inactive friends. The power to influence the neighbors is what we model as influence probability. The problem we tackle in this paper is how to learn influence probabilities among the users, by mining the available set of past propagations. Formally, we want to learn a function $p : E \rightarrow [0, 1] \times [0, 1]$ assigning to both directions of each edge $(v, u) \in E$ the probabilities: $p_{v,u}$ and $p_{u,v}$.

4. SOLUTION FRAMEWORK

In the following, we introduce the framework we adopt which is an instance of the General Threshold Model. Consider an inactive user u and the set of its activated neighbors S, and suppose that each neighbor $v \in S$ activates after v and u became neighbors. To predict whether u will activate, we need to determine $p_u(S)$, the joint influence probability of S on u. If $p_u(S) \geq \theta_u$, where θ_u is the activation threshold of user u, we can conclude that u activates. For ease of exposition, assume individual probabilities of influence between users are static, i.e., are independent of time. We do not need this assumption for our results. Since influence probabilities are meant for use in viral marketing [9] [8], our definitions must be consistent with the diffusion models used in these papers. These

[2]At the time of our submission, this paper was not yet published. We thank the reviewers for pointing it to us.

[3]For convenience, we assume social ties are never broken. This assumption is inessential.

papers typically assume the diffusion models are *monotone*, which says the function $p_u(S)$ should satisfy: $p_u(S) \leq p_u(T)$ whenever $S \subseteq T$. Moreover, it should be *submodular*, i.e., $p_u(S \cup \{w\}) - p_u(S) \geq p_u(T \cup \{w\}) - p_u(T)$ whenever $S \subseteq T$. There can be various ways to define $p_u(S)$. In this paper, for computational ease, we assume that the probability of various friends influencing u are independent of each other. Hence, the joint probability $p_u(S)$ can be defined as follows:

$$p_u(S) = 1 - \prod_{v \in S} (1 - p_{v,u}) \tag{1}$$

In the context of testing a learned model, we need to be able to compute and update the influence probabilities on the fly. That is, as new neighbors get activated, the joint influence probability needs to be updated. We should be able to compute $p_u(S \cup \{w\})$ incrementally without revisiting the neighbor set influence probabilities, i.e., solely in terms of $p_u(S)$ and $p_{w,u}$. This is not part of the requirement imposed by the diffusion models assumed for viral marketing. Thus, this is a desirable, but not mandatory property.

THEOREM 1. *The joint influence probability as defined in Eq. 1 is monotone and submodular. Besides, it can be updated incrementally if the individual influence probabilities $p_{v,u}$ are static.*

Proof. Let S be the set of neighbors of u that are active and suppose a new neighbor w of u gets activated. The new joint influence probability $p_u(S \cup \{w\})$ can be computed incrementally from $p_u(S)$ as follows.

$$\begin{aligned} p_u(S \cup \{w\}) &= 1 - (1 - p_{w,u}) * \prod_{v \in S} (1 - p_{v,u}) \\ &= 1 - (1 - p_{w,u}) * (1 - p_u(S)) \\ &= p_u(S) + (1 - p_u(S)) * p_{w,u} \end{aligned} \tag{2}$$

The monotonicity can be seen from Eq. 2. The difference $p_u(S \cup \{w\}) - p_u(S)$ is clearly non-negative as the domain of individual probabilities is $[0, 1]$. Similarly, submodularity can be shown as follows.

$$\begin{aligned} &p_u(S \cup \{w\}) - p_u(S) - p_u(T \cup \{w\}) + p_u(T) \\ &\quad = (1 - p_u(S)) * p_{w,u} - (1 - p_u(T)) * p_{w,u} \\ &\quad = (p_u(T) - p_u(S)) * p_{w,u} \geq 0 \end{aligned}$$

since, by monotonicity, $p_u(T) \geq p_u(S)$. □

User Influenceability. As mentioned in the introduction, there can be three reasons which prompt any user to perform an action. First, influence from friends and family members. Second, she is affected by some external event(s). And the last is that she is a very active user and is doing things without getting influenced by anyone.

In this work, we mainly focus on modeling and learning the influence propagation from neighbors. For some users, external influence plays a significant role and for others that is not the case. Users who are initiators of actions and who are more influenced by external factors are unpredictable or less influenceable. So, we define an *influenceability score* representing how influenceable a user is, as the ratio between the number of actions for which we have evidence that the user was influenced, over the total number of actions performed by the user. More precisely we define:

$$infl(u) = \frac{|\{a \mid \exists v, \Delta t : prop(a, v, u, \Delta t) \wedge 0 \leq \Delta t \leq \tau_{v,u}\}|}{A_u} \tag{3}$$

In the equation above, we can use any appropriate value for the parameter $\tau_{v,u}$. We propose to use the average time delay, defined as follows:

$$\tau_{v,u} = \frac{\sum_{a \in \mathbf{A}} (t_u(a) - t_v(a))}{A_{v2u}} \tag{4}$$

where $t_u(a)$ is the time when u performs a and $\mathbf{A}$ is the set of actions in the training data. We conjecture that users with a high value for $infl(u)$ may exhibit a high degree of being influenced by their neighbors compared to those with a low value for this metric.

Action Influenceability. We define the *influence quotient* for an action to distinguish between actions for which there is more evidence of influence propagation from the rest of the actions. More precisely, we define:

$$infl(a) = \frac{|\{u \mid \exists v, \Delta t : prop(a, v, u, \Delta t) \wedge 0 \leq \Delta t \leq \tau_{v,u}\}|}{\text{number of users performing } a} \tag{5}$$

We expect that for actions with high $infl(a)$, predictions (based on influence models) of user performing those actions will yield a relatively higher precision and recall values compared to other actions. We will revisit this in the experimental section.

5. MODELS

Recall from the previous section that we assume the probabilities of influence by individual neighbors of a user are independent. Thus, if we have a model for capturing individual influences, we can compute the joint influence using Eq. 1. Next, we propose 3 types of models to capture $p_{v,u}$, the probability with which u is influenced by its neighbor v. The first class of models assumes the influence probabilities are static and do not change with time. The second class of models assumes they are continuous functions of time. As a preview, it will turn out continuous time models are by far the most accurate, but they are very expensive to test on large data sets. Thus, we propose an approximation known as Discrete Time Models where the joint influence probabilities can be computed incrementally and thus efficiently. In all the models we propose, we also discuss how to learn estimates of various parameters from the training data set.

5.1 Static Models

These models are independent of time and are the simplest to learn and test. We present three instances of static models.

Bernoulli distribution. Under this model, any time a contagious user v tries to influence its inactive neighbor u, it has a fixed probability of making u activate. If u activates, it is a successful attempt. Each attempt, which is associated with some action, can be viewed as a Bernoulli trial. The Maximum Likehood Estimator (MLE) of success probability is the ratio of number of successful attempts over the total number of trials. Hence, influence probability of v on u using MLE is estimated as:

$$p_{v,u} = \frac{A_{v2u}}{A_v} \tag{6}$$