Learning Influence Probabilities in Networks: A Review

Navreet Kaur*
Indian Institute of Technology, Delhi
2015TT10917
tt1150917@iitd.ac.in

Sanyam Gupta*
Indian Institute of Technology, Delhi
2015EE30761
ee3150761@iitd.ac.in

***Abstract*— This paper presents a survey of the work done in the field of Social Network Analysis with a special focus on learning influence probabilities of propagation models. Since almost all the propagation models for maximizing the spread of influence assume the influence probabilities as an input, it is essential to develop a scalable, accurate and efficient method to learn the latent representations of the nodes and edges in the network.**

***Index Terms*— Data Mining, Social Networks, Social Influence, Representation Learning, Influence Probabilities, Propogation Models**

I. INTRODUCTION

In the recent years, with the advent of social networks and micro blogs such as Facebook and Twitter, research on propagation models has exploded in the recent years. This is not only due to the availability of huge data sets for research purposes but also because studies on the spread of influence in a society are important from economic, political and social perspective and find applications in viral marketing, outbreak detection, finding trendsetters, information feed ranking, etc.

With the power of social influence, a company can market a new product by first convincing a small number of influential users to adopt the product and then triggering further adoptions through the effect of word of mouth (also referred to as influence maximization). The propagation model governs how influence diffuses or propagates through the network. The two fundamental propagation models are Independent Cascade Model (IC) and Linear Threshold Model (LT). In both these models, at a given timestamp, each node is either active or inactive, and each nodes tendency to become active increases monotonically as more of its neighbors become active.

While a rich body of the research on influence propagation models is devoted to representation learning, influence maximization and analysis of propagation of information through networks, we focus our study on methods of learning influence probabilities of these models. Most of the studies on the analysis of influence propagation assume that the edge weights of the social network graph i.e. the influence probabilities are given to them.

Saito et al. [6] studied how to learn the probabilities for the IC model from a set of past propagations. They formalize this as a likelihood maximization problem and then apply the expectation maximization (EM) algorithm to solve it. Goyal et al. [2] also studied the problem of learning influence probabilities. They focus on the time varying nature of influence, and on factors such as the influenceability of a specific user, and influence-proneness of a certain action.

In this paper, we focus on approaches of learning influence probabilities in propagation models by building on the existing aforementioned approaches.

*Both the authors are first authors

II. MOTIVATION

Social influence has become a prevalent, yet complex force that drives our social decisions, making a clear need for methodologies to characterize, understand, and quantify the underlying mechanisms and dynamics of social influence.

In the majority of the literature on influence maximization, the edge-weighted social graph is assumed as input to the problem, without addressing the question of how the probabilities are obtained. Building models to make predictions on the action taken by a user requires us to know the influence probabilities first. From where do these probabilities come from, how can they be quantified based on network parameters, action log of users and other features, and how can they be computed for a real world social network are very important questions that need to be addressed and form the primary focus of this paper.

III. PROBLEM FORMULATION

In this section, we introduce some terminology and then formulate the problem of learning influence probabilities of propagation models.

Definition 1 (Social Network) An undirected network $G = (V, E, \tau)$, where V is a set of nodes representing users or other entities, $E \subseteq V \times V$ denotes a set of edges that represent social relations between two nodes u and v and $\tau : E \rightarrow N$ is a function corresponding to the time-stamp at which the edge was created i.e. $e_{u,v} \neq 0$.

Definition 2 (Action Log) Users perform actions in a network represented by the relation $\alpha = (U, A, T)$, which indicates that user $u \in U$ performed an action $a \in A$ at time-stamp $t_u \in T$. Users in A correspond to the nodes in G. A denotes the universe of actions.

Definition 3 (Propagation Graph) For $a \in A$, a propagation graph $P(a) = (V(a), E(a))$ is a set of nodes $V(a) = \{v | \exists t : (v, a, t) \in \alpha\}$ and set of edges $E(a) = \{e_{u,v} | e_{u,v} \in E; \exists (u, a, t_u), (v, a, t_v) \in \alpha; t_u < t_v, \tau(u, v) \leq t_u\}$.

The propagation graph connects the users that performed any action with each other such that $u \rightarrow v$ if u's action influenced v's action. Note that $P(a)$ cannot contain cycles due to the time constraint and thus, is a DAG. Node u is called the *parent* of v if $e_{u,v} \in E(a)$ and node u is the *child* of v if $e_{v,u} \in E(a)$. Nodes are said to be *active* if they have been influenced with information.

Problem 1 Model the power of a node to influence its neighbours i.e. predict influence probabilities of the network $G(V, E, \tau)$ such that each $e_{u,v} \in E$ equals the probability $p_{u,v}$ representing the strength of influence exerted by u on v; if no link exists between u and v, then $e_{u,v} = 0$. Formally, maximize the probability that a user u will take action a at time $t + \delta t$ given the propagation model $P(a)$ and action logs α i.e.

$$\prod_{i=1}^{N}(P(\alpha(u, a, t + \delta t)|\alpha(U, A, T), P(a)) \tag{1}$$

for N instances of the action log.
The following paper compares and analyses work done so far on tackling the problem of learning influence probabilities among the users by mining set of past information diffusion, using user-specific features and network structure etc.

IV. RELATED WORK

For the influence maximization problem, selecting the initial nodes which would maximize the influence spread in a network depends on the influence estimates of those nodes. The state of the influence graph is modelled using propagation models whose influence probabilities must be learned first. In this section, we discuss and compare various approaches that have been taken to address the problem of learning influence probabilities of a user in a network. Mainly two approaches have been discussed: (i) Learning parameters of propagation models which correspond to the influence probabilities, and (ii) Deep learning approaches which do not require propagation models but directly learn the influence probabilities given the network.

A. *Learning Probabilistic Models*

We first describe some fundamental types of propagation models (Linear Threshold, Independent Cascade) that simulate information diffusion through a network and then move on to the approaches to solving the problem of learning the influence probabilities represented by parameters of these models. In each of the following two models, a node is either active or inactive at a given time-step. Time unfolds in discrete steps as the propagation proceeds and a node's tendency to become active increases monotonically as more of its neighbours become active.
Linear Threshold (LT) Model: Each node u has been influenced by each neighbour v with a weight $P_{u,v}$ such that $\sum_{\forall v \in V} P_{u,v} \leq 1$. Node u becomes active if at time-step t, $\sum_{\forall v \in V} P_{u,v} >= \theta_u$, where θ_u is a threshold picked uniformly at random from the interval $[0, 1]$ for u being activated.

Independent Cascade (IC) Model: Each link (u, v) has a *diffusion probability* $\lambda_{u,v}$ associated with it. When a node u becomes active at time-step t, it has a single chance of influencing its inactive neighbour v with the probability $\lambda_{u,v}$ independent of the history and v becomes active at time-step $t + 1$ if u succeeds. Node u cannot make any further attempts to activate v independent of whether or not it was able to activate v earlier. If multiple neighbours of v become active at time-step t, then the activation sequences are all performed at same time t but sequences in an arbitrary fashion.
Like most propagation models, since these models also require the parameters representing diffusion probabilities to be specified in advance, it is essential to learn them. This problem was first addressed by Saito *et. al*[1] by introducing a *Continuous-Time Independent (CTIC) Model* and solving the learning problem as well by formulating it as a Likelihood-Maximization problem and using Expectation-Maximization to solve it. They propose a broader framework that simultaneously generalizes the LT and IC model and has equivalent formulations in terms of thresholds and cascades.

Continuous-Time Independent (CTIC) Model: Each link (u, v) has *time-delay parameter* $r_{u,v} > 0$ and *diffusion parameter*$0 < k_{u,v} < 1$ associated with it. Unlike in LT and IC, the time unfolds continuously as the propagation proceeds. If a node u is active at time t, it has a single chance to activate any neighbour v if it was inactive before time $t+\delta$ with $k_{u,v}$ probability, where δ is the delay-time chosen from an exponential distribution with $r_{u,v}$ as the parameter. Same as in IC model, u cannot make any further attempts at activating v whether or not it succeeded earlier.
Since the time-delay parameter $\mathbf{r} = (r_{u,v})_{(u,v)\in E}$ and diffusion parameter $\mathbf{k} = (k_{u,v})_{(u,v)\in E}$ are not available, these need to be learned from the past information propagation histories, defined as $\mathcal{D}_m = \{D_m; m = 1, ..., M\}$. Likelihood function $\mathcal{L}(\mathbf{r}, \mathbf{k}; \mathcal{D}_m)$ to estimate $\mathbf{r}$ and $\mathbf{k}$ from $\mathcal{D}$ is defined as follows:

$$\mathcal{L}(\mathbf{r}, \mathbf{k}; \mathcal{D}_m) = \prod_{m=1}^{M}(\prod_{t\in\tau_m}\prod_{v\in D_m(t)} h_{m,v} \prod_{v\in D_m}\prod_{w\in F(v)\setminus D_m} g_{m,v,w}) \tag{2}$$

where
$g_{m,v,w}$ is the probability hat the node w is ot activated by the node v within the observed time period $[t_m, T_m]$,
$h_{m,v}$ is the probability density that the node v is activated at time $t_{m,v}$,
τ_m is the observation-time list,
$F(v)$ is the set of childs of v.

An EM like estimation method is proposed to estimate $\mathbf{r}$ and $\mathbf{k}$. Initial estimates $\bar{\mathbf{r}}$ and $\bar{\mathbf{k}}$ are considered and then time-delay and diffusion parameters are iteratively updated in a way to maximize the likelihood, guaranteeing the

convergence.

While the formulation done for CTIC model is elegant, it is not scalable to large data sets. In the subsequent section, we also discuss the reasons for these limitations. Goyal *et. al*[2] proposed an extension of the General Threshold Model to learn influence probabilities.

General Threshold Model: Each node is active or inactive at a given time-stamp and it's tendency to become active increases monotonically as more of its neighbours become active. In the General Threshold Model, each node has a monotone activation function that maps the set of neighbours to a real value. If this value becomes greater than threshold, the node becomes active.

Another approach for learning influence probabilities is suggested by Goyal *et. al* in [2], wherein they propose a solution framework based on the General Threshold Model. The assumption is made that the probabilities of influence of individual neighbours are independent of each other. Hence, the joint probability can be calculated as given in (2).

$$p_u(S) = 1 - \prod_{v \in S}(1 - p_{v,u}) \tag{3}$$

where, $p_{v,u}$ is probability by which u is influenced by neighbour v.

He proposes three types of models which we will discuss below.

Static model: These are the simplest models and are independent of time. Three instances of static models are discussed in the paper-

Bernoulli Distribution: In this model, a user v has a fixed probability of making an inactive neighbour u activate.

Jaccard Index: It is defined by ratio of the size of intersection and size of union of two sample sets.

Partial Credit: It is possible that a user u is influenced to perform an action by the combination of neighbours who have performed that action before. We can say that each of the predecessors share 'credit' for influencing u.

Continous Time (CT) model: This model takes into consideration the fact that influence probability may not remain constant independently of time. It defines $p_{v,u}^t$ as follows:

$$p_{v,u}^t = p_{v,u}^0 e^{-(t-t_v)/\tau_{v,u}} \tag{4}$$

This class of models also allows us to predict the timestamp at which the user is most likely to perform the action.

Discrete Time model: Because Continuous Time models are computation time, an incremental approximation in the form of Discrete Time Models is used.

We also discuss some of the algorithms presented in [3] for learning the parameters of the various models. These are given in Fig. 1 and 2.

Algorithm 1 Learning - Phase1

```
for each action a in training set do
  current_table = φ
  for each user tuple < u, a, t_u > in chronological order do
    increment A_u
    parents = φ
    for each user v : (v, a, t_v) ∈ current_table && (v, u) ∈ E^{t_v} do
      if t_u > t_v then
        increment A_{v2u}
        update τ_{v,u}
        insert v in parents
      increment A_{v&u}
    for each parent v ∈ parents do
      update credit_{v,u}
    add (u, a, t_u) to current_table
```

Fig. 1. Algorithm 1

Algorithm 2 Learning - Phase2

```
for each action a in training set do
  current_table = φ
  for each user tuple < u, a, t_u > in chronological order do
    parents = φ
    for each user v : (v, a, t_v) ∈ current_table && (v, u) ∈ E^{t_v} do
      if 0 < t_u − t_v < τ_{v,u} then
        increment A_{v2u}
        insert v in parents
    for each parent v ∈ parents do
      update credit_{v,u}^{τ_{v,u}}
    if parents != φ then
      update infl(u)
    add (u, a, t_u) in current_table
```

Fig. 2. Algorithm 2

B. Neural Network approaches: DeepInf

Proposed in [3], DeepInf is a deep learning based framework in the domain of representation learning that effectively projects influence dynamics and network structures into a latent space. It automatically discovers hidden signals in social influence that increase its predictive power.

Given an ego user v, social influence prediction has been formulated as a binary graph classification problem that is solved by minimizing the negative log likelihood objective function given by (5).

$$\mathcal{L}(\Theta) = -\sum_{i=1}^{N} log(P_\Theta(s_{v_i}^{t_i+\delta t} | G_{v_i}^r, S_{v_i}^{t_i})) \tag{5}$$

Here, G_v^r is the subnetwork (induced by random walk with restart upto a fixed number of nodes) of the set of nodes whose shortest distance from v is less than r. This is the input for the neural network model. S_v^t is the set of action statuses of neighbours sampled in G_v^r.

The proposed neural network end-to-end approach outputs latent representation of v, which can then be used to predict her action status. It consists of a network embedding layer, an instance normalization layer, an input layer, several graph

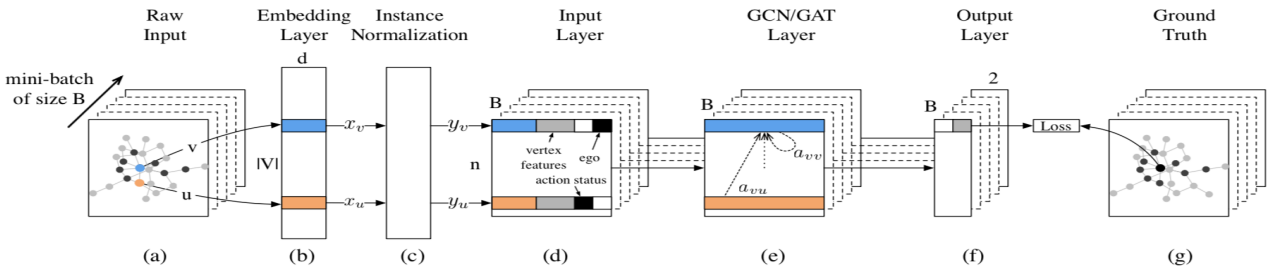

Fig. 3. DeepInf Framework model

convolutional or graph attention layers, and an output layer all of which have been explained below. The complete model is depicted in Figure 3.

The **embedding layer** encodes network structure properties into a lower dimensional space. They have used pre-trained embeddings to transform each user into a *D*-dimensional representation.

The **instance normalization layer** removes mean and variance specific to instances and helps focusing on relative positions of users rather than absolute ones.

The **input layer** creates the feature vector for training. Apart from the low-dimensional embedding, two binary variables are concatenated to make the feature vector. The first variable indicates user's actions status and the second indicated whether the user is an ego user.

Graph Convolutional Network(GCN) based encoding used multiple GCN layers stacked together to carry out a non-linear transformation. It takes an input $\boldsymbol{H}$ which contains features of every vertex and applies non-linear transformation to output H'.

Multi-head Graph Attention(GAT) inculcates the attention mechanism into GCN. Attention coefficients for each pair of vertices is computed if they form an edge or self loop which allows us to capture the graph structural information better. Multi-head attention uses an aggregation function on the outputs of K single attention layers to output H'.

The **output layer** creates a 2-dimensional representation for each user. We then optimise the log-likelihood loss described in (5) after comparison with the ground truth.

V. LIMITATIONS OF EXISTING WORK

In this section, we briefly dicuss the limitations of approaches discussed in the previous section and also describe the reasoning for the same.

- Saito *et. al*[1] gave the CTIC model and formalized a EM method to maximize the objective function for finding the time-delay and diffusion parameters of the model given the past information propagation history. Though elegant, their theoretical formulation has certain limitations and the method is not scalable for large datasets. This is attributed to the updation of influence probabilities associated with each edge at every step of the EM algorithm. Also, they assume that the input propagations have the same shape as generated by the IC model. Pre-processing is needed to meet the gap between the real-world data and the data required for this model. This is due to the fact that the input consists of sets of users activated in corresponding **discrete** time-steps in the IC propagation but CTIC considers them to be continuous and hence, the need for pre-processing.
- Models proposed in Goyal et.al[2] make assumptions made in a Viral Marketing framework, which ignores the effect of time and assumes edges have constant influence probabilities as labels. However, time varying nature of influence is very common and needs to be taken into account as well. For users with low influencibility might 'attenuate' some of the incoming influence from the neighbours. Such real-world phenomenon need to be accounted for.
- DeepInf is also limited in its approach because it only accepts vertex-level features. It is expected that it will learn the r-ego network on its own.

REFERENCES

[1] K. Saito, M. Kimura, and K. Ohara. Learning Continuous-Time Information Diffusion Model for Social Behavioral Data Analysis

[2] Goyal, A. , Bonchi, F and Lakshmanan, L. V. S. 2010. Learning Influence Probabilities In Social Networks In ACM WSDM. In Proceedings of the 1th Advances in Machine Learning, First Asian Conference on Machine Learning, ACML 2009, pp. 322-337, Nanjing, China, November, 2009

[3] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. DeepInf: Social Influence Prediction with Deep Learning

[4] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. 2009. Social influence analysis in large-scale networks In KDD 09. ACM, 807816.

[5] A. Najar, L. Denoyer, and P. Gallinari. Predicting information diffusion on social networks with partial knowledge In WWW 12 Companion, pages 11971204, New York, NY, USA, 2012. ACM

[6] K. Saito, R. Nakano, and M. Kimura. Prediction of information diffusion probabilities for independent cascade model In Proc. of the 12th Int. Conf. on Knowledge-Based Intelligent Information and Engineering Systems (KES08), 2008.

[7] N. Phan, D. Dou, B. Piniewski, and D. Kil Social restricted Boltzmann machine: Human behavior prediction in health social networks In Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining, Aug. 2015, pp. 424431

[8] K. Kutzkov, A. Bifet, F. Bonchi, and A. Gionis Strip: stream learning of influence probabilities In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 275283, ACM, 2013

[9] L.Liu, J.Tang, J.Han,and S.Yang Learning influence from heterogeneous social networks In Data Mining and Knowledge Discovery, 25(3):511544, 2012