DeepIS: Susceptibility Estimation on Social Networks

Anonymous
anonymous@anonymous.edu
Anonymous University
Anonymous city, Anonymous country

ABSTRACT

Influence diffusion estimation is a crucial problem in social network analysis. Most prior works mainly focus on predicting the total influence spread, i.e., the expected number of influenced nodes given an initial set of active nodes (aka. seeds). However, accurate estimation of susceptibility, i.e., the probability of being influenced for the individual, is more appealing and challenging in real-world applications. Previous methods generally adopt the Monte Carlo simulation or heuristic rules to estimate the influence, resulting in high computational cost or unsatisfactory estimation error when these methods are used to estimate susceptibility. In this work, we propose to leverage graph neural networks (GNNs) for predicting susceptibility. As GNNs aggregate multi-hop neighbor information and could generate over-smoothed representations, the prediction quality for susceptibility is undesirable. To address the shortcomings of GNNs for susceptibility estimation, we propose a novel `DeepIS` model with a two-step approach: (1) a coarse-grained step where we estimate each node's susceptibility with GNNs; (2) a fine-grained step where we aggregate neighbors' coarse-grained susceptibility estimations to compute the fine-grained estimate for each node. We conduct extensive experiments and show that on average `DeepIS` achieves five times smaller estimation error than state-of-the art GNN approaches and two magnitudes faster than the Monte Carlo simulation.

CCS CONCEPTS

• **Information systems** → **Social networks**; **Social advertising**.

KEYWORDS

influence estimation, graph neural networks, influence maximization

ACM Reference Format:
Anonymous. 2018. DeepIS: Susceptibility Estimation on Social Networks. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Woodstock '18, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Social network users continuously generate massive contents, which could reach large audiences through network propagation/influence. The prevalence of social networks expedites many social influence-based applications, e.g., viral marketing [16, 31] and recommendation [29, 38]. One essential problem is to predict the social influence. For example, a company may dispatch free products to some users as *seeds* and be interested in the number of users affected by the seeds through the *word-of-mouth* effect, i.e., they concern *how many users will be influenced in total* [16]? However, in many real-world scenarios, predicting the probability of being influenced for targeted individuals (or **susceptibility**) is more appealing. In targeted advertising, the advertisers may consider the susceptibility of males/females, a certain age group, and other relevant information to craft the best marketing strategy [15]. In political campaigns, influential ads for specific targets can significantly affect the election results [12]. Furthermore, susceptibility estimation could serve as a core component for a wide-known problem in social network studies: *Influence Maximization (IM)* [16], which aims to select a seed set that could influence the most users.

Many existing studies have proposed diffusion models to characterize and analyze social influence, e.g., the Independent Cascade (IC) model [8, 9] and the Linear Threshold (LT) model [11]. In this work, we focus on IC model for susceptibility estimation due to its popularity in the literature [5, 16, 19, 44], but our proposed approach can be adapted for other diffusion models. It has been proved that, computing the exact influence spread, i.e., the expected number of influenced users, under IC/LT model is #P-hard [5, 7]. Hence, most previous efforts use the Monte Carlo (MC) simulation to estimate the influence spread, i.e., repeating the diffusion process and averaging the results [10, 16, 20, 27, 44]. However, the MC simulation is extremely time-consuming and cannot be scaled to large networks. Thus, some studies propose heuristic algorithms to approximate the influence spread, e.g., based on the node degree [6], the shortest path [5, 14] and others [14, 44]. However, the existing methods focus on the influence spread and overlook the problem of estimating susceptibility for specific individuals or groups. According to our experiments in Section 5, adopting existing methods to estimate susceptibility can lead to high computational costs or unsatisfactory estimation error.

In this paper, to our best knowledge, we conduct the first systematic study on susceptibility estimation. The estimation problem can be regarded as a regression task on nodes. Thereby, we propose to tackle the problem from the graph learning perspective, leveraging the graph neural networks (GNNs). There are two major challenges in susceptibility estimation with GNNs. First, the influence diffuses through many users. To tackle the susceptibility for a specific node u, we should consider the multi-hop neighbors of u. Meanwhile, GNNs aggregate multi-hop neighbors' information by stacking model layers, and each layer aggregates one-hop neighbors' information [17]. However, it has been shown that multiple stacked layers can lead to over-smoothed node representations [23] and tend to hurt the performance of any prediction task on individual nodes [41]. Second, GNNs aggregate neighbors' representations using uniform weight [17] or attention mechanism [37]. Thus, the aggregation schemes of GNNs do not capture the characteristics of the underlying diffusion model.

To address the aforementioned challenges, we propose a novel `DeepIS` model for susceptibility estimation. We integrate the characteristics of the diffusion model into `DeepIS` with a two-step approach. In the first step, we construct features by encoding the seed nodes into a multi-hot vector and taking the power sequence of the influence probability matrix. The features are then fed into a multi-layer GNN for coarse-grained susceptibility estimations. We adopt independent estimation on every node to eliminate the aggregation step from neighbors and thus circumvent the problem of over-smoothed representations of GNNs. In the second step, to enable fine-grained computation, we propose a propagation scheme to diffuses each node's estimation among its neighbors. We devise the propagation equation motivated by the concept of stationary distribution of the random walk process. Since the influence spread is intrinsically different with the random walk, we devise an iterative propagation scheme considering the specific characteristics of the diffusion model. We note that `DeepIS` is *inductive*, i.e., we could train the model on one graph and run on other graphs, which avoids expensive retraining on large datasets and enables real-time influence analysis.

We verify the effectiveness and efficiency of `DeepIS` on several public graph datasets. Experimental results reveal that `DeepIS` provides five times smaller estimation error compared with state-of-the-art GNN models and reduces about two orders of magnitude running time compared with the MC simulation. Furthermore, `DeepIS` performs well as a subroutine of the IM problem. For convenient reproduction of the results, we release the implementation of `DeepIS` [1].

2 RELATED WORKS

In this section, we review the related literature on influence spread estimation as well as GNNs.

2.1 Influence Spread Estimation

Influence spread estimation approximates the expected number of influenced nodes given a seed set, and it is a core component in IM algorithms [5, 6, 16, 20]. Computing the influence spread exactly under IC /LT model is #P-hard [5, 7].Hence, the MC simulation is adopted to estimate the influence spread [10, 20, 27, 44]. Some recent works adopt the sampling of reverse reachable (RR) sets to estimate the influence spread [1, 35, 36]. RR sets-based estimation can be regarded as a specific MC method. To alleviate the computational overhead of the MC simulation, [6] proposes to utilize the node's outdegree and a discount rule to approximate the marginal influence spread. [14] approximates the marginal influence spread of a single node by iteratively computing a system of linear equations.

[1]https://github.com/Anonymous-Projects-Maker1/DeepIS

Another line of works utilizes GNNs to predict the influence spread or the node states. [21, 30] utilize the neighbor nodes' states and the local subgraph to predict a node's activation state. [3, 22, 43] aim to predict the popularity of social contents based on the cascade information in a time window. They conduct predictions based on message cascade logs. Note that this line of works does not consider the influence diffusion models. Furthermore, all existing works focus on predicting the total number of influenced nodes without considering the susceptibility of individual nodes, which are not suitable for targeted influence analysis studied in this work.

2.2 Graph Neural Networks (GNNs)

GNNs are getting surged attention in recent years. As deep neural networks have achieved widespread success in the Euclidean data space, GNNs mainly focus on the non-Euclidean graph structures. The general paradigm of GNNs alternates between node feature transformation and neighbor nodes' information aggregation. For a k-layer GNN, a node aggregates information within k-hop neighbors. Specifically, the k-th layer transformation is

$$a_v^{(k)} = \text{AGGREGATE}^k\left(\left\{h_u^{(k-1)}, u \in \mathcal{N}(v)\right\}\right) \tag{1}$$

$$h_v^{(k)} = \text{COMBINE}^k\left(\left\{h_v^{(k-1)}, a_v^{(k)}\right\}\right) \tag{2}$$

where $a_v^{(k)}$ represents node v's aggregated feature vector from neighbors, and $h_v^{(k)}$ is the k-th layer feature vector. The flexibility of AGGREGATE and COMBINE function induces different GNN models. The high-level representations of nodes or graphs are utilized for different tasks. GNNs have been applied in various learning tasks related to graphs, e.g., semi-supervised node classification [17, 24, 32], link prediction [42], and social cascade prediction [4, 43].

Among existing works in GNNs, our work is closely related to the PPNP algorithm proposed in [18] and the MONSTOR algorithm proposed in [19]. PPNP separates the feature transformation and information propagation into independent stages and computes an approximate stationary distribution of the personalized page rank (PPR) [28] to propagate each node's prediction to other nodes on the graph. The propagation scheme is as follows:

$$\begin{aligned} Z^{(0)} &= f_\theta(X) \\ Z^{(k)} &= (1-\alpha)\hat{\tilde{A}}Z^{(k-1)} + \alpha Z^{(0)} \\ Z^{(K)} &= \text{softmax}\left((1-\alpha)\hat{\tilde{A}}Z^{(K-1)} + \alpha Z^{(0)}\right) \end{aligned} \tag{3}$$

where f_θ is a multi-layer nonlinear function, X is the initial feature matrix, α is the teleport probability in PPR, and $\hat{\tilde{A}}$ is the symmetrically normalized adjacency matrix with self-loops of the graph. PPNP motivates our work as we propose to combine the characteristics of the diffusion models and the iterative propagation process in a two-stage framework.

MONSTOR focuses on the incremental influence spread estimation of nodes under IC model based on a stacked structure of multiple GNNs. However, the model weights the neighbor node's features according to the influence strength in the aggregation stage, ignoring the characteristics of diffusion models. Furthermore, the estimation error accumulates with the number of GNNs increases.

In this work, we estimate the susceptibility of each node using a single neural network and propagate the estimations leveraging the characteristics of the diffusion model to obtain precise estimations.

3 PROBLEM FORMULATION

In this section, we introduce the diffusion model and the problem formulation for susceptibility estimation. For ease of presentation, we focus on IC model but our proposed framework can be adapted to support other diffusion models like LT model.

3.1 The IC model

For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set, with $|\mathcal{V}| = n$. Each $e_{ij} \in \mathcal{E}$ is associated with an influence probability p_{ij}, indicating the probability of node i influence node j in the next step, if i is influenced in the current step. We denote P as the influence strength matrix with $P_{n\times n} = (p_{ij})_{n\times n}$. Under IC model, the influence propagates from the initial seed nodes S at timestep 0. At each timestep $t > 0$, the influenced (or activated) node i in timestep $t-1$ propagate influence to their out-neighbor j with probability p_{ij}. Note that each activated node only has one chance to activate its neighbors and remains to be in the active state. The propagation process terminates at timestep t if is are no newly activated node.

3.2 The Susceptibility Estimation Problem

Given IC model, we use D to represent an diffusion instance and denote the state of node i on instance D by $I_{S,D}(n_i)$. If n_i is influenced by seed S in instance D , $I_{S,D}(n_i) = 1$ and 0 otherwise. The susceptibility of node i is represented by $\zeta_S(n_i) = \mathbb{E}_{D\sim p(D)}[I_{S,D}(n_i)]$. The *Susceptibility Estimation Problem* aims to estimate $\zeta_S(n_i)$ for each node n_i. We denote the influence spread of seeds S as $\sigma(S)$. One can see that $\sigma(S) = \sum_{n_i \in \mathcal{V}} \zeta_S(n_i)$.

4 DEEPIS

In this section, we propose the DeepIS model. DeepIS consists of two stages, i.e., (1) construct features to feed a GNN for coarse-grained susceptibility estimation of each node. (2) propagate estimations, which propagates each node's estimated susceptibility to neighbors through an iterative process. Finally, we compute the loss between the estimations after these procedures with the training labels, to optimize the model in an end-to-end manner. Note that we refer to the feature construction and coarse-grained computation as the *GNN stage*, while the estimation propagation as the *propagation stage*. We demonstrate the framework of the DeepIS model in Fig. 1. We describe the two stages in the following sections.

4.1 GNN Stage

Feature construction. Given the influence strength matrix P and the initial seed nodes. We can represent the seed nodes S as a multi-hot vector $\boldsymbol{x}$, with $\boldsymbol{x}[i] = 1$, if $n_i \in S$, otherwise $\boldsymbol{x}[i] = 0$.

The matrix P preserves both the graph topology and the influence strengths, while $\boldsymbol{x}$ preserves the seeds information. We denote the node's susceptible probability vector as $\boldsymbol{\zeta}_S$, i.e., $\boldsymbol{\zeta}_S = [\zeta_S(n_1), \zeta_S(n_2), ...]^T$. Theoretically, $\boldsymbol{\zeta}_S$ and the influence spread $\sigma(S)$ are completely determined by P and $\mathbf{x}$.

Since P and $\boldsymbol{x}$ involve sufficient information for computing the objective $\boldsymbol{\zeta}_S$ and $\sigma(S)$, we consider leverage P and $\boldsymbol{x}$ to construct the features. We first introduce the following definition.

Definition 4.1 (k-step influence probability). A node's k-step influence probability is the activation probability at timestep k.

Note that $(P^T)^k\boldsymbol{x}$ is an upper bound of the k-step influence probability of all nodes under IC model [44], since $(P^T)^k$ includes all paths from an arbitrary seed to an arbitrary target node with length k. Based on this observation, we utilize the matrix power of P^T and $\boldsymbol{x}$ to construct the **feature matrix** X for all nodes, i.e.,

$$X = [\boldsymbol{x}, P^T\boldsymbol{x}, (P^T)^2\boldsymbol{x}..., (P^T)^k\boldsymbol{x}] \tag{4}$$

Node estimation. Once the feature matrix X is obtained, we adopt a nonlinear neural function f_θ to predict each node's $\zeta_S(n_i)$, where θ is the function parameter. In this work, we adopt a two-layer fully connected neural network as the f_θ. Our studies show that the simple f_θ provides reasonable results and computation effectiveness. Specifically, we calculate the estimations of all node as follows:

$$\begin{aligned} f_\theta(X) &= \text{Sigmoid}(\text{Relu}(XW_1 + \boldsymbol{b}_1)W_2 + \boldsymbol{b}_2) \\ \hat{\boldsymbol{y}}_i &= f_\theta(X[i,:]) \end{aligned} \tag{5}$$

Note that the choice of the function f_θ is flexible. We may integrate other prediction models to substitute the f_θ adopted above, e.g., GCNs [17] or attention-based graph models [37].

4.2 Propagation Stage

We propose the propagation scheme that draws the intuition from the random walk. We first describe the stationary distribution of random walk process and analyze the difference between random walk and random influence diffusion. Then we propose the propagation scheme for DeepIS. Finally, we conduct some theoretical analysis on the convergence property of the propagation process.

Propagation scheme. Previous works utilize the stationary distribution of random walk to propagate information to neighbor nodes [18]. Meanwhile, the diffusion process has differences compared against the random walk. For the random walk, the probability of a node being visited in a given timestep is the summation of its neighbor's probabilities multiplying the transition matrix. The stationary distribution $\boldsymbol{\pi}$ satisfies the following equation [18]:

$$\boldsymbol{\pi} = P\boldsymbol{\pi} \tag{6}$$

For the random walk with restart probability α, the stationary distribution matrix $\boldsymbol{\Pi}$ satisfies (7), [18].

$$\boldsymbol{\Pi} = (1-\alpha)P\boldsymbol{\Pi} + \alpha \boldsymbol{I}_n \tag{7}$$

In the random walk, a node can be passed for multiple times, while in influence diffusion, once a node is influenced, it cannot be influenced twice. The influence spread process is intrinsically different with random walk.

Furthermore, under IC model, the probability of a node not being influenced is the product of not being influenced by any neighbor. Therefore, we propose the following *approximate stationary distribution* $\tilde{\boldsymbol{\zeta}}$ of the random diffusion, which satisfies the relation (8).

$$\tilde{\zeta}_i = 1 - \prod_{n_j \in \mathcal{N}(n_i)} (1 - P_{ji}\tilde{\zeta}_j) \tag{8}$$

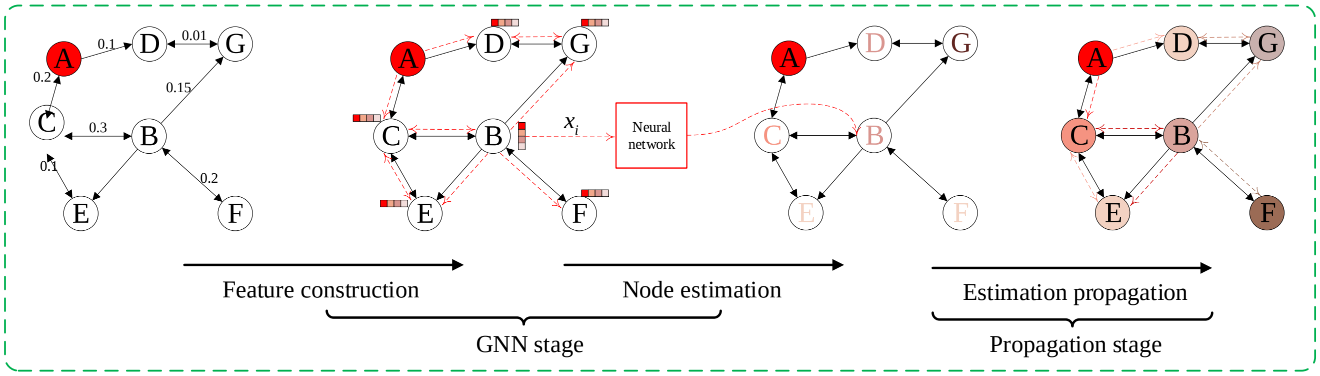

Figure 1: The framework of the proposed DeepIS model for susceptibility estimation. Firstly, the model constructs each node's feature vector according to influence strengths and the seed nodes (the red node). Secondly, a nonlinear neural function maps each node's feature vector to a probability in $[0,1]$. Finally, each node's result propagates to its neighbor nodes except the seed nodes to obtain the final estimation.

$\mathcal{N}(n_i)$ in (8) denotes the in-neighbors of node n_i. Equation (8) means that for node n_i, the approximate susceptible probability $\tilde{\zeta}_i$ is influenced by its in-neighbors, and $\tilde{\zeta}_i$ will also influence its out-neighbor's approximate susceptible probabilities. We note that, to adapt our approach for other diffusion models, one only needs to update Equation (8). For example, we can design the approximate susceptible probability for LT model as:

$$\tilde{\zeta}_i = \sum_{n_j \in \mathcal{N}(n_i)} P_{ji}\tilde{\zeta}_j \tag{9}$$

Next we define the propagation function $g(\cdot)$ and describe the propagation scheme. $g(\cdot)$ is defined as follows:

$$g(\boldsymbol{x}) = \begin{pmatrix} g(x_1) \\ g(x_2) \\ \cdots \\ g(x_n) \end{pmatrix} = \begin{pmatrix} 1 - \prod_{n_j \in \mathcal{N}(n_1)} (1 - P_{j1}x_j) \\ 1 - \prod_{n_j \in \mathcal{N}(n_2)} (1 - P_{j2}x_j) \\ \cdots \\ 1 - \prod_{n_j \in \mathcal{N}(n_n)} (1 - P_{jn}x_j) \end{pmatrix} \tag{10}$$

The propagation scheme is formulated in (11), which originates from the relation in (8).

$$\begin{aligned} \hat{\boldsymbol{y}}^{(0)} &= \hat{\boldsymbol{y}} \\ \hat{\boldsymbol{y}}^{(1)} &= g(\hat{\boldsymbol{y}}^{(0)}) \\ \hat{\boldsymbol{y}}^{(q)} &= g(\hat{\boldsymbol{y}}^{(q-1)}) \end{aligned} \tag{11}$$

Note that q denotes the iteration number.

Note that $g(x_i)$ is determined by in-neighbors of node n_i, hence one iteration essentially propagates each node's information into its one-hop neighbors. After q iterations of propagation, we output the $\hat{\boldsymbol{y}}^{(q)}$ as the final results.

The AGGREGATE and COMBINE in (12) summarize the propagation stage.

$$\begin{aligned} a_i &= \text{AGGREGATE}\left(\{x_j, j \in \mathcal{N}(n_i)\}\right) = g(x_i) \\ h_i &= \text{COMBINE}\left(\{a_i, x_i\}\right) = \begin{cases} a_i, & n_i \in S \\ 1, & n_i \notin S \end{cases} \end{aligned} \tag{12}$$

Note that (12) shows that for seed nodes, we do not combine the neighbor information since the susceptibility is always 1.

Propagation analysis In the following, we give the fixed point existence and the convergence analysis of the iteration function $g(\cdot)$, to study the choice of iteration number q. Note that a fixed point $\boldsymbol{x}^*$ satisfies $\boldsymbol{x}^* = g(\boldsymbol{x}^*)$. We first introduce the following fixed point theorem.

THEOREM 1 ([2]). *Let $D = \{(x_1, \cdots, ..., x_n)^T | a_i \le x_i \le b_i\}$, for some collection of contents of $a_1, \cdots, a_n$ and $b_1, \cdots, b_n$. Suppose $g(\cdot)$ is a continuous function from $D \subset \mathbb{R}^n$ into $\mathbb{R}^n$ with property that $g(\boldsymbol{x}) \in D$ whenever $\boldsymbol{x} \in D$. Then $g(\cdot)$ has a fixed point in D.*

Suppose that all the component functions of $g(\cdot)$ have continuous partial derivatives and a constank $K \le 1$ exists with

$$\left|\frac{\partial g_i(\boldsymbol{x})}{\partial x_j} \le \frac{K}{n}\right|, \quad \forall \boldsymbol{x} \in D$$

for all $j = 1, \cdots, n$, and all component function $g_i(\cdot)$. Then the sequence $\{\boldsymbol{x}^{(k)}\}_{k=0}^{\infty}$ defined by a $\boldsymbol{x}^{(0)} \in D$ and generated by

$$\boldsymbol{x}^{(k)} = g(\boldsymbol{x}^{k-1}), \quad \forall k \ge 1$$

converges to the unique fixed point $\boldsymbol{p} \in D$.

Based on the Theorem 1, we present the following propositions.

PROPOSITION 4.2. *$g(\cdot)$ in (10) has a fixed point in $[0,1]^n$.*

PROOF. Let $D = [0,1]^n$ and $P_{ij} \in [0,1], \forall i, j$. One can see that $g(\cdot)$ is continuous. For $\forall \boldsymbol{x} \in D$, we have $g(x_i) \in [0,1], \forall i$. Hence we have $g(\boldsymbol{x}) \in D$, i.e., $g(\cdot)$ has a fixed point in D. □

PROPOSITION 4.3. *The iteration process in (11) may not converge for an arbitrary influence probability matrix P.*

Proof. The partial derivatives of $g_i(\boldsymbol{x})$ w.r.t x_j is computed as:

$$\begin{aligned}\frac{\partial g_i(\boldsymbol{x})}{\partial x_j} &= \frac{\partial\left(1-\prod_{n_k\in\mathcal{N}(n_i)}(1-P_{ki}x_k)\right)}{\partial x_j}\\ &= \frac{\partial\left(1-\prod_{k=1\to n}(1-P_{ki}x_k)\right)}{\partial x_j}\\ &= \frac{\partial\left(1-(1-P_{ji}x_j)\prod_{k=1\to n,k\neq j}(1-P_{ki}x_k)\right)}{\partial x_j}\\ &= P_{ji}\prod_{k=1\to n,k\neq j}(1-P_{ki}x_k)\end{aligned}$$

Since $x_k \in [0,1]$, we have $\frac{\partial g_i(\boldsymbol{x})}{\partial x_j} \in [P_{ji}\prod_{k=1\to n,k\neq j}(1-P_{ki}), P_{ji}]$. For an arbitrary matrix P, the condition $P_{ji} \leq \frac{1}{n}$ may not be satisfied. Hence the iteration process in (11) may not converge. □

Propositon 4.2 and 4.3 reveal that the propogation iteration process has a fixed point. However, the iteration process may not converge for an arbitrary graph influence probability matrix P. Hence we should not set the iteration number q too large. In practice, an iteration number $q \leq 5$ leads to satisfactory results in experiments, and we will study the iteration effect in section 6.4.

4.3 End-to-end Training

Since the propagation function $g(\cdot)$ is differentiable, we could integrate the propagation scheme into model training and utilize the final output $\hat{\boldsymbol{y}}^{(q)}$ to compute loss. Note that we omit the superscript (q) in the following for conciseness.

We adopt the mean square error on training cascades considering the probability estimation error on each node, i.e.,

$$L_{\text{MSE}} = \frac{1}{M}\sum_{m=1}^{M}\|\hat{\boldsymbol{y}}_{(m)} - \boldsymbol{y}_{(m)}\|_2 \tag{13}$$

where M is the number of training cascades, i.e., the training set $T = |\{C_1, C_2, \cdots, C_M\}|$. Each cascade $C = \{\boldsymbol{x}, \boldsymbol{y}\}$ includes the initial multi-hot seed vector $\boldsymbol{x}$ and the groundtruth target probability vector $\boldsymbol{y}$.

We also measure the influence spread error leveraging the mean relative square error (MRSE). The MRSE reflects the graph-level estimation performance, with optimization robustness and differentiability.

$$L_{\text{MRSE}} = \frac{1}{M}\sum_{m=1}^{M}\left(\frac{\|\hat{\boldsymbol{y}}_{(m)}\|_1 - \|\boldsymbol{y}_{(m)}\|_1}{\|\boldsymbol{y}_{(m)}\|_1}\right)^2 \tag{14}$$

We describe the final objective to be optimized as follows:

$$\text{Loss} = \underbrace{L_{\text{MSE}}}_{\text{node lever error}} + \underbrace{\lambda L_{\text{MRSE}}}_{\text{graph lever error}} + \underbrace{\gamma\|\theta\|_2}_{\text{regularization term}} \tag{15}$$

which consists of three components, with hyperparameters λ and γ. The L2 regularization term aims to mitigate the over-fitting and facilitates the convergence process.

Table 1: Dataset statistics. Avg. p is the average probability.

Dataset	Type	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Avg. p
CORA-ML	Citation	2810	7981	0.1168
CITESEER	Citation	2110	3668	0.1152
PUBMED	Citation	19717	44321	0.1155
MS-ACADEMIC	Co-author	18333	81894	0.1157

5 EXPERIMENTAL SETUP

In this section, we compare our method with several state-of-the-art GNN models and the MC simulation under several evaluation metrics to measure the prediction performance. We also report the performance of `DeepIS` when used as a subroutine in IM algorithms. Note that we implement all algorithms with python3.7 and run on a server with a Intel Xeon E5-2620 CPU.

5.1 Datasets

We adopt four commonly used datasets: CORA-ML [25], CITESEER [33], PUBMED [26], and MS-ACADEMIC [34]. For each dataset, we select the largest connected component for our experiments. Since the groundtruth influence propagation probabilities between nodes are unavailable on these datasets, previous works generate these parameters manually [5, 39]. Similarly, we generate these influence parameters from a discrete distribution [0.001, 0.01, 0.05, 0.1, 0.15, 0.2, 0.3] with uniform probabilities. We summarize the statistics of datasets in Table 1.

For each dataset, we generate the propagation cascades according to the following two strategies. Firstly, we generate seed sets with the size of 50, 100, 150, 200, 250, 300. For each seed size, we generate 100 random seed vector $\boldsymbol{x}$. We repeat 10000 MC simulations for each seed set to obtain the groundtruth susceptible probability vector $\boldsymbol{y}$. Each cascade instance C consists of the $\boldsymbol{x}$ and the $\boldsymbol{y}$, i.e., we construct C as a $|\mathcal{V}| \times 2$ matrix, with $C[:,0] = \boldsymbol{x}$ and $C[:,1] = \boldsymbol{y}$. We randomly sample 80% of these cascades as the train set, 10% as the validation set and 10% as the test set. Secondly, to avoid small influence probabilities of random seeds, we further compute the top-K nodes with the highest degrees as seed sets for each seed size as test sets. We will compare the model's performance on two kinds of seed sets respectively.

5.2 Baselines

We compare the following GNN approaches with `DeepIS`.

- GCN [17]. GCN utilizes the normalized Laplacian matrix to average each node's feature with neighbor's features, and transform these averaged features with a shared parameter matrix.
- GraphSAGE[13]. GraphSAGE aggregates each node's neighbor features and concatenates the aggregated features with node's self-features. GraphSAGE transforms node features in an inductive manner and is applicable on an unseen graph, differing from that of GCN.
- GAT [37]. GAT improves the performance of GraphSAGE by introducing the attention mechanism. The attention mechanism calculates aggregation weights between nodes in the aggregation layer, introducing flexibility and directionality for different nodes.

- SGC [40]. SGC simplifies previous GCN-like models by propagating node features in a preprocessing stage, then predicting each node's target value with a simple logistic regression. The SGC follows a similar philosophy with our work to some extent, i.e., separating the propagation and prediction.
- MONSTOR [19]. MONSTOR aims to estimate the total influence spread of a seed set by stacking multiple GCN models. One GNN model predicts the incremental influence spread of one-step influence propagation based on former GNN model's outputs.

Transductive vs Inductive. For the transductive methods, including GCN and SGC, we train on four datasets. For the inductive methods, including our DeepIS, GraphSAGE, GAT, and MONSTOR, we train on the CORA-ML and test on the other three datasets.

Once the independent susceptible probabilities are obtained, the total influence spread is easily calculated by summation. Since the influence spread calculation is a key component in the IM problem, we further compare the influence spread and running time of two representative IM algorithms, i.e., UBLF [44], and CELF [20], when invoking the MC simulation as the subroutine and invoking DeepIS model as the subroutine. We denote each scenario as UBLF_{MC} (CELF_{MC}) and $\text{UBLF}_{\text{DeepIS}}$ ($\text{CELF}_{\text{DeepIS}}$), respectively.

5.3 Implementation details

For GCN and GraphSAGE, we adopt a 2-layer structure, as reported in their original papers [13, 17]. For GAT, we set the number of attention heads to 4 and the dimension of each attention channel to 8 [37]. For SGC, we set the iteration number in the preprocessing stage to 2, as reported in the original work. For MONSTOR, We set the number of stacks to 3, and for each stack, we adopt a 2-layer GCN network, as reported in [19]. For the proposed DeepIS model, we leverage a 2-layer MLP as the fucntion $f(\cdot)$, as stated in section 4.1. For the feature dimension $k+1$ in feature construction stage and the iteration number q in propagation stage, we vary k in $[2,5]$ and vary q in $[2,5]$. We will discuss the effects of these important hyperparameters in Section 6.4. Note that the number of hidden units for all the baseline models and the DeepIS is set to 64. We provide the identical feature matrix constructed using Equation (4) for all models for fair comparison.

5.4 Evaluation metrics

We adopt several metrics to evaluate the performance of different methods thoroughly. Note that T denotes the number of test instances.

5.4.1 Mean Error (ME). The *mean error* measures the average susceptibility estimation error on each node.

$$\text{ME} = \frac{1}{T}\sum_{t=1}^{M}\frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}|\hat{\boldsymbol{y}}_i^{(t)} - \boldsymbol{y}_i^{(t)}| \tag{16}$$

5.4.2 Total Error (TE). The *total error* measures the average influence spread estimation error on the graph level.

$$\text{TE} = \frac{1}{T}\sum_{t=1}^{T}\left(\left|\sum_{i=1}^{|\mathcal{V}|}\hat{\boldsymbol{y}}_i^{(t)} - \sum_{i=1}^{|\mathcal{V}|}\boldsymbol{y}_i^{(t)}\right|\right) \tag{17}$$

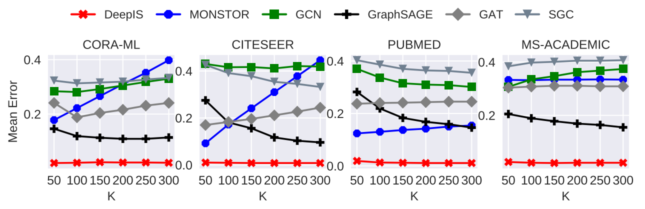

Figure 2: The mean error of different methods on all datasets with random seeds(IC).

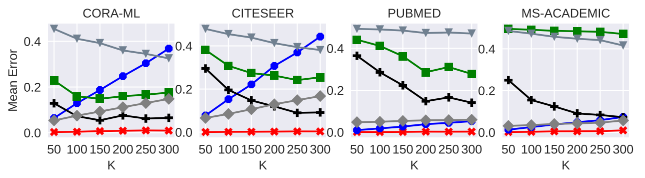

Figure 3: The mean error of different methods on all datasets with random seeds(LT).

5.4.3 Running Time (RT). We compare the average running time between the $\text{UBLF}_{\text{DeepIS}}$ ($\text{CELF}_{\text{DeepIS}}$) and the UBLF_{MC} (CELF_{MC}). The RT metric basically reveals the speed comparison between DeepIS and MC.

5.4.4 Influence Spread (IS). We compare the influence spread performance between the $\text{UBLF}_{\text{DeepIS}}$ ($\text{CELF}_{\text{DeepIS}}$) and the UBLF_{MC} (CELF_{MC}). The IS metric evaluates the proposed DeepIS's performance when running as a subroutine in IM algorithms.

6 EXPERIMENTAL RESULTS

In this section, we firstly compare the ME and TE of all methods on four datasets. Besides, we substitute MC in two IM algorithms with DeepIS to compare the efficiency and influence spread. We analyze the effect of several hyperparameters in the model further. Finally, we show the effect of the propagation stage intuitively.

6.1 Estimation error on random seeds

The mean error of all methods on IC and LT models are demonstrated in Fig. 2 and Fig. 3, respectively. Fig. 2 demonstrates the mean error of all methods on different datasets with varying seed set sizes. We can see that other GNN methods perform unsatisfactorily for this problem. GraphSAGE achieves the smallest estimation error in most scenarios among the GNN baselines. However, the mean error of GraphSAGE exceeds 0.1, which is a relatively large deviation from the groundtruth. In all scenarios, DeepIS achieves the smallest mean error compared with the baselines. The estimation error of DeepIS is stable and fluctuates around 0.02 for different scenarios, which is on average five times smaller than baselines.

For the total error in the graph level, Fig. 4 and Fig. 5 show the results of all methods with different seed sizes on IC and LT models, respectively. We can see that the results of total error generally align with the mean error. While there exist mismatches since the total error is not strictly equal to the summation of absolute errors of all nodes. For example, on the CORA-ML dataset under IC model in Fig. 4, other models achieve the best total error and

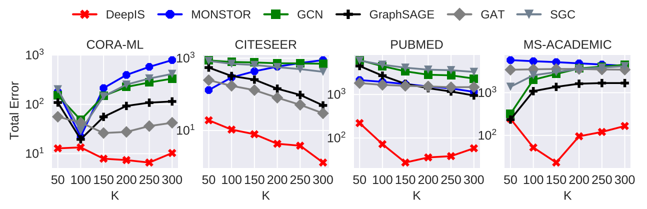

Figure 4: The total error of different methods on all datasets with random seeds(IC).

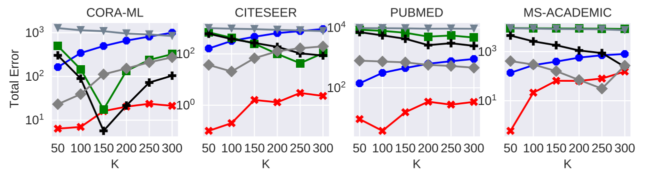

Figure 5: The total error of different methods on all datasets with random seeds(LT).

are adjacent to DeepIS with seed size 100. However, in Fig. 2, their mean errors are much larger than that of DeepIS, revealing that their predictions are inaccurate on most nodes, and the relatively small total errors are obtained due to the cancellation of errors when summing estimations across different nodes.

6.2 Estimation error on high degree seeds

For high-degree seeds, Fig. 6 and Fig. 7 demonstrates the mean error on IC and LT models, respectively. Fig. 8 and Fig. 9 show the total error on two models, respectively. High degree seeds are more likely to induce high susceptible probabilities than randomly selected seeds. In our experiments, the mean susceptible probability is larger than 0.15 for these datasets. We can see that, in Fig. 6 and Fig. 7, DeepIS remains to produce the smallest mean error compared with other baselines. The results are similar with those of Fig. 2 and Fig. 3. The mean error of DeepIS fluctuates between 0.01 and 0.02, while other baselines are larger than 0.1. Fig. 8 and Fig. 9 show that in most settings, DeepIS obtains competitive performance for the total error. Some baselines achieves smaller total error under some settings. However, as stated in Sec. 6.1, the smaller total errors are achieved by the cancellation of different node's errors.

The results in Sections 6.1 and 6.2 reveals, DeepIS achieves on average five times smaller mean error compared with the baselines. We interpret the inferior performance of the baseline GNN models as two-fold: First, most of these models are devised for the node classification problem, while estimating the susceptible probability for each node is more challenging and complicated than the classification problem. Second, each node's target value is determined by the underlying propagation model, while only the proposed DeepIS considers the specific diffusion characteristics of the propagation in the model construction and inference.

6.3 Influence maximization

In this section, we report the model's performance when substituting MC with DeepIS in IM algorithms on the CORA-ML dataset. Fig. 10 and Fig. 11 illustrates the influence spread comparison on

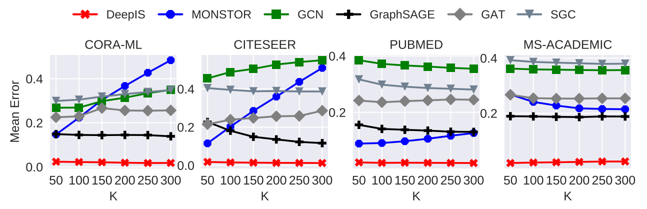

Figure 6: The mean error of different methods on all datasets with high degree seedes(IC).

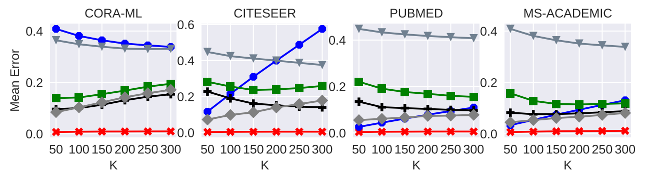

Figure 7: The mean error of different methods on all datasets with high degree seedes(LT).

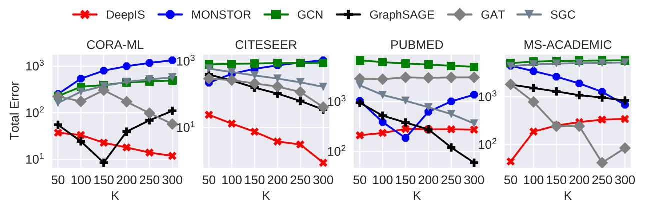

Figure 8: The total error of different methods on all datasets with high degree seedes(IC).

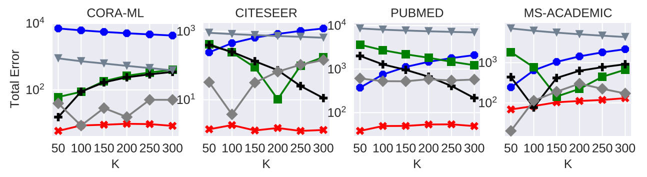

Figure 9: The total error of different methods on all datasets with high degree seedes(LT).

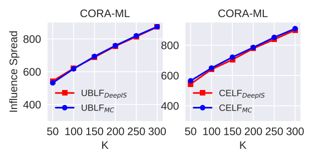

Figure 10: IS (IC).

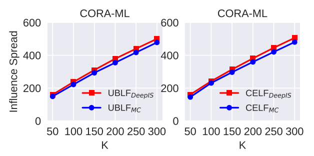

Figure 11: IS (LT).

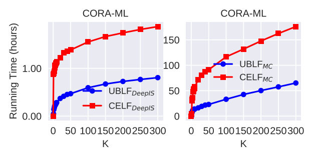

Figure 12: RT (IC).

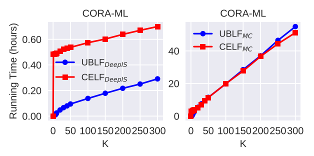

Figure 13: RT (LT).

Table 2: The hyperparameters

Hyperparameter	Description	Values
k	Feature dimension	2, 3, **4**, 5
q	Iteration number	1, **2**, 3, 4
λ	Weight of L_{MRSE}	**1e-4**, 1e-3, 1e-2, 1e-1
γ	Weight of $\|\theta\|_2$	1e-5, **1e-4**, 1e-3, 1e-2

IC and LT models, respectively. The influence spread of computed seeds with each size is calculated using MC with 10000 repeats. For IC model in Fig. 10, we can see that the UBLF$_{\mathrm{DeepIS}}$ achieves almost identical influence spread with that of UBLF$_{\mathrm{MC}}$ for each seed size. The results are similar for CELF$_{\mathrm{DeepIS}}$ and CELF$_{\mathrm{MC}}$. For LT model in Fig. 11, the final influence spread of UBLF$_{\mathrm{DeepIS}}$ is also close with that of UBLF$_{\mathrm{MC}}$. The influence spread comparison between CELF$_{\mathrm{DeepIS}}$ and CELF$_{\mathrm{MC}}$ is similar. Generally, the results in Fig. 10 and Fig. 11 reveal that, substituting MC with DeepIS has minor effects on the quality of seeds calculated by IM algorithms.

Fig. 12 and Fig. 13 illustrates the running time comparison on IC and LT model, respectively. For IC model in Fig. 12, UBLF$_{\mathrm{DeepIS}}$ consumes about 0.75h to obtain 300 seeds, while UBLF$_{\mathrm{MC}}$ requires about 70 hours. CELF$_{\mathrm{DeepIS}}$ consumes about 1.9h for 300 seeds while CELF$_{\mathrm{MC}}$ requires about 180 hours. For LT model, the results are similar, UBLF$_{\mathrm{DeepIS}}$ and CELF$_{\mathrm{DeepIS}}$ requires less than 0.8h, while UBLF$_{\mathrm{MC}}$ and CELF$_{\mathrm{MC}}$ requires about 60 hours. Comparing the left with the right for both diffusion models in Fig. 12 and Fig. 13, we can see that algorithms invoking DeepIS reduces about two magnitudes of running time compared with their counterparts invoking MC.

6.4 Hyperparameter analysis

In this section, we analyze the effects of several import hyperparameters of the DeepIS model. We list the information of hyperparameters in Table 2. We conduct experiments on the CORA-ML dataste under IC model.

Fig. 14 shows the effects of k. We can see that larger k achieves lower mean error since larger k provides more information of the propagation. However, larger k results in higher computational cost. Hence, we set the default k to 4 to balance effectiveness and efficiency. Fig. 15 illustrates the effect of q. The results reveal that $q = 2$ and $q = 3$ outperforms other settings. The results in Fig. 15 reveals that we do not need to iterate too many times (less than 5) in the propagation stage. We set $q = 2$ as the default value considering diverse scenarios. Fig. 16 shows small λ performs well for the mean error metric, since small λ forces the model to focus on the node-level error. Hence, we choose λ = 1e-4 as the default value. Fig. 17 adjusts the weight of L2 regularization. We can see that it is suitable to set γ in the range [1e-5, 1e-3], since γ =1e-2 results in obviously larger mean error than other settings. The reason may lie in the underfitting of the model for a large L2 penalty. Hence, we set γ =1e-4 as the default value.

6.5 Effect of Two-Stage Approach

In this section, we study the effectiveness of the additional stage that performs the fine-grained propagation in DeepIS, by comparing the estimation performance between with and without the propagation stage. We randomly select 300 seeds on the CORA-ML dataset and

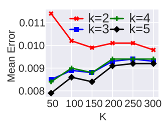

Figure 14: k

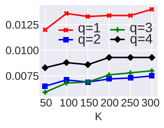

Figure 15: q

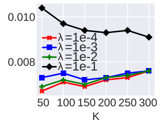

Figure 16: λ

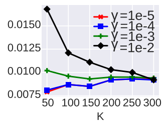

Figure 17: γ

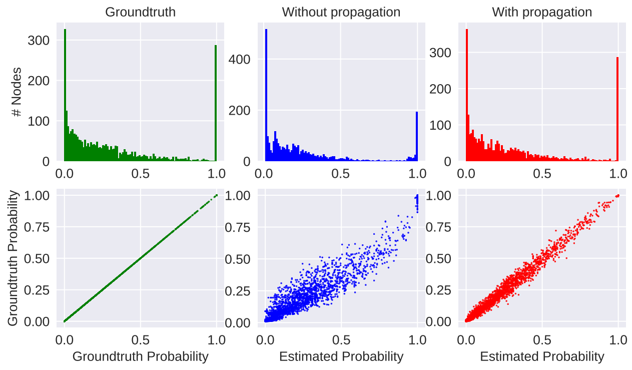

Figure 18: The estimation performance comparison of DeepIS between with and without fine-grained propagation.

compute the susceptible probabilities of all nodes using MC under IC model as the groundtruth. To illustrate the comparison results intuitively, we plot the histograms and estimation-groundtruth pairs in Fig. 18. The first row of Fig. 18 compares the histograms of groundtruth, estimation without propagation, and estimation with propagation. The horizontal axis represents the susceptible probabilities and the vertical axis represents the number of nodes with the same probability. The second row shows the relation between estimation and target values, with the horizontal axis representing the estimated susceptible probabilities and the vertical axis representing the groundtruth. Comparing the second column with the third column in Fig. 18, we can see that the propagation step reduces the estimation error for most nodes effectively. The scatter points in the third column are highly concentrated around the ideal green line in the first column.

7 CONCLUSION

In this paper, we propose the DeepIS model to estimate the susceptible probability of independent network individuals effectively. We consider this problem as a node regression task in the graph learning perspective. We proposed the DeepIS model combining the GNN structures and the characteristics of the influence diffusion model. The DeepIS model predicts the susceptible probability of each node independently and calculates the influence spread by summing up all node's estimations . The propagation scheme explicitly leverages the diffusion properties. Experimental results reveal that the DeepIS model predicts both each node's susceptible probability and the total influence spread accurately. The model also performs well as a subroutine in IM algorithms. In the future, we devote to extend the method under more general diffusion models, e.g. continuous-time models, and scenarios when the network topology is partially observed.

REFERENCES

[1] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing social influence in nearly optimal time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 946–957.

[2] Richard L Burden and J Douglas Faires. [n.d.]. Numerical Analysis 7th edition, 2001. *Thomson Learning ISBN 0-534-38216-9* ([n. d.]).

[3] Qi Cao, Huawei Shen, Keting Cen, Wentao Ouyang, and Xueqi Cheng. 2017. DeepHawkes Bridging the Gap between Prediction and Understanding of Information Cascades. *CIKM* (2017), 1149–1158.

[4] Qi Cao, Huawei Shen, Jinhua Gao, Bingzheng Wei, and Xueqi Cheng. 2020. Popularity prediction on social platforms with coupled graph neural networks. *WSDM 2020 - Proceedings of the 13th International Conference on Web Search and Data Mining* (2020), 70–78. arXiv:1906.09032

[5] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1029–1038.

[6] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2009), 199–207.

[7] Wei Chen, Yifei Yuan, and Li Zhang. 2010. Scalable influence maximization in social networks under the linear threshold model. In *Proceedings - IEEE International Conference on Data Mining, ICDM*. 88–97.

[8] Jacob Goldenberg, Barak Libai, and Eitan Muller. 2001. Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth. *Marketing Letters* 12, 3 (2001), 211–223.

[9] Jacob Goldenberg, Barak Libai, and Eitan Muller. 2001. Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata. *Academy of Marketing Science Review* 2001, January 2001 (2001), 1.

[10] Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. 2011. CELF++: Optimizing the greedy algorithm for influence maximization in social networks. *Proceedings of the 20th International Conference Companion on World Wide Web, WWW 2011* (2011), 47–48.

[11] Mark Granovetter. 1978. Threshold models of collective behavior. *American journal of sociology* 83, 6 (1978), 1420–1443.

[12] Katherine Haenschen and Jay Jennings. 2019. Mobilizing Millennial Voters with Targeted Internet Advertisements: A Field Experiment. *Political Communication* 36, 3 (2019), 357–375.

[13] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. , 59 pages.

[14] Kyomin Jung, Wooram Heo, and Wei Chen. 2012. IRIE: Scalable and robust influence maximization in social networks. *Proceedings - IEEE International Conference on Data Mining, ICDM* (2012), 918–923. arXiv:1111.4795

[15] Kai Kaspar, Sarah Lucia Weber, and Anne-Kathrin Wilbers. 2019. Personally relevant online advertisements: Effects of demographic targeting on visual attention and brand evaluation. *PloS one* 14, 2 (2019), e0212419.

[16] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 137–146.

[17] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* (2016), 1–14. arXiv:1609.02907

[18] Johannes Klicpera, Aleksandar Bojchevski, and Stephan G¨unnemann. 2019. Predict then propagate:Graph neural networks meet personalized pagerank. *7th International Conference on Learning Representations, ICLR 2019 - Conference Track Proceedings* (2019), 1–14.

[19] Jihoon Ko, Kyuhan Lee, Kijung Shin, and Noseong Park. 2020. MONSTOR: An Inductive Approach for Estimating and Maximizing Influence over Unseen Social Networks. (2020). arXiv:2001.08853 http://arxiv.org/abs/2001.08853

[20] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne Vanbriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 420–429.

[21] Carson K Leung, Alfredo Cuzzocrea, Jiaxing Jason Mai, Deyu Deng, and Fan Jiang. 2019. Personalized DeepInf: enhanced social influence prediction with deep learning and transfer learning. *2019 IEEE International Conference on Big Data (Big Data)* (2019), 2871–2880.

[22] Cheng Li, Jiaqi Ma, Xiaoxiao Guo, and Qiaozhu Mei. 2017. DeepCas: An end-to-end predictor of information cascades. *26th International World Wide Web Conference, WWW 2017* (2017), 577–586. arXiv:1611.05373

[23] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)* 24, 20 (2018), 64.

[24] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S.Zemel. 2019. LanczosNet: Multi-Scale Deep Graph Convolutional Networks. *7th International Conference on Learning Representations, ICLR 2019 - Conference Track Proceedings* (2019), 1–15.

[25] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.

[26] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U M D EDU. 2012. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, Vol. 8.

[27] Naoto Ohsaka, Takuya Akiba, Yuichi Yoshida, and Ken-ichi Kawarabayashi. 2014. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[28] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The pagerank citation ranking: Bringing order to the web.* Technical Report.

[29] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2110–2119.

[30] Jiezhong Qiu, Jie Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Jian Tang. 2018. DeepInf: Social influence prediction with deep learning. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2018), 2110–2119. arXiv:1807.05560

[31] Matthew Richardson and Pedro Domingos. 2002. Mining Knowledge-Sharing Sites for Viral Marketing. (2002), 61–70.

[32] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: towards deep graph convolutional network on node classification. *8th International Conference on Learning Representations, ICLR 2020 - Conference Track Proceedings* 2019 (2020), 1–17.

[33] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93.

[34] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[35] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Vol. 2015-May. Association for Computing Machinery, New York, New York, USA, 1539–1554.

[36] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 75–86. arXiv:1404.0900

[37] Petar Veličković, Arantxa Casanova, Pietro Liò, Guillem Cucurull, Adriana Romero, and Yoshua Bengio. 2018. Graph attention networks. *6th International Conference on Learning Representations, ICLR 2018* (2018), 1–12. arXiv:1710.10903

[38] Hongyang Wang, Qingfei Meng, Ju Fan, Yuchen Li, Laizhong Cui, Xiaoman Zhao, Chong Peng, Gong Chen, and Xiaoyong Du. 2020. Social Influence Does Matter: User Action Prediction for In-Feed Advertising. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 246–253.

[39] Zheng Wen, Branislav Kveton, Michal Valko, and Sharan Vaswani. 2017. Online influence maximization under independent cascade model with semi-bandit feedback. In *Advances in Neural Information Processing Systems*, Vol. 2017-Decem. 3023–3033. arXiv:1605.06593

[40] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying graph convolutional networks. *36th International Conference on Machine Learning, ICML 2019* 2019-June (2019), 11884–11894. arXiv:1902.07153

[41] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken Ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. *35th International Conference on Machine Learning, ICML 2018* 12 (2018), 8676–8685. arXiv:1806.03536

[42] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems* 2018-Decem, NeurIPS (2018), 5165–5175. arXiv:1802.09691

[43] Qingyuan Zhao, Murat A. Erdogdu, Hera Y. He, Anand Rajaraman, and Jure Leskovec. 2015. SEISMIC: A self-exciting point process model for predicting tweet popularity. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Vol. 2015-Augus. 1513–1522.

[44] Chuan Zhou, Peng Zhang, Jing Guo, Xingquan Zhu, and Li Guo. 2013. UBLF: An upper bound based approach to discover influential nodes in social networks. In *Proceedings - IEEE International Conference on Data Mining, ICDM*. IEEE, 907–916.