Active Learning of Model Parameters for Influence Maximization

Tianyu Cao[1], Xindong Wu[1], Tony Xiaohua Hu[2], and Song Wang[1]

[1] Department of Computer Science, University of Vermont, USA
[2] College of Information Science and Technology, Drexel University, USA

Abstract. Previous research efforts on the influence maximization problem assume that the network model parameters are known beforehand. However, this is rarely true in real world networks. This paper deals with the situation when the network information diffusion parameters are unknown. To this end, we firstly examine the parameter sensitivity of a popular diffusion model in influence maximization, i.e., the *linear threshold model*, to motivate the necessity of learning the unknown model parameters. Experiments show that the influence maximization problem is sensitive to the model parameters under the linear threshold model. In the sequel, we formally define the problem of finding the model parameters for influence maximization as an active learning problem under the linear threshold model. We then propose a weighted sampling algorithm to solve this active learning problem. Extensive experimental evaluations on five popular network datasets demonstrate that the proposed weighted sampling algorithm outperforms pure random sampling in terms of both model accuracy and the proposed objective function.

Keywords: Influence maximization, Social network analysis, Active Learning.

1 Introduction

Social networks have become a hot research topic recently. Popular social networks such as Facebook and Twitter are widely used. An important application based on social networks is the so-called "viral marketing", the core part of which is the influence maximization problem [5, 11, 9].

A social network is modeled as a graph $G = (V, E)$, where V is the set of users (nodes) in the network, and E is the set of edges between nodes, representing the connectivity and relationship of users in that network. Under this model, the influence maximization problem in a social network is defined as *extracting a set of k nodes to target for initial activation such that these k nodes yield the largest expected spread of influence, or interchangeably, the largest diffusion size (i.e., the largest number of nodes activated), where k is a pre-specified positive integer*. Two information diffusion models , i.e., the *independent cascade model* (IC model) and the *linear threshold model* (LT model) are usually used as the underlying information diffusion models. The influence maximization problem has been investigated extensively recently [3, 2, 4, 1, 14].

To the best of our knowledge, all previous algorithms on influence maximization assume that the model parameters (i.e., diffusion probabilities in the IC model and

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part I, LNAI 6911, pp. 280–295, 2011.

thresholds in the LT model) are given. However, this is rarely true in real world social networks. In this paper we relax this constraint for the LT model, assuming that the model parameters are unknown beforehand. Instead, we propose a framework of active learning to obtain those parameters. In this work, we focus on learning the model parameters under the LT model since it is relatively simple, and we will investigate the same problem under the IC model in the future.

Learning the information diffusion models has been studied in [7, 13, 12]. However, there is a problem with the methods from [7, 13, 12], as these methods assume that a certain amount of information diffusion data (propagation logs in [7]) on the network is available. This data is usually held by the social network site and not immediately available to outsiders. In some cases, it is not available at all due to privacy considerations. Considering a scenario in which we wish to use "viral marketing" techniques to market some products on Facebook, most likely we cannot get any data of information diffusion on Facebook due to privacy reasons.

Therefore, we need to actively construct the data of information diffusion in order to learn the information diffusion model parameters. This naturally falls into the framework of active learning. There are two advantages using the active learning approach. Firstly we are no longer restricted by the social network sites' privacy terms. Secondly we have the additional advantage that we can explicitly control what social influence to measure. For example, we can restrict the influence scope to either "music" or "computer devices". Based on the scope, we can learn the diffusion model with a finer granularity. Therefore we can do influence maximization on the "music" products and the "computer device" products separately. Intuitively, the influential nodes of "music" products and the "computer devices" should be different.

A simple way to construct the information diffusion data is to send free products to some users and see how their social neighbors react. All social neighbors' reaction constitutes the information diffusion data. This is our basic idea to acquire the diffusion data. Now we rephrase this process in the context of information diffusion. We set some nodes in a network to be active. Then we observe which nodes become active at the following time steps. The observed activation sequences can be used as the information diffusion data. We can then make inference of the model parameters based on this observed activation sequences.

In this context, we would naturally want to achieve the following two goals: (1) we would like to send as few free products as possible to learn the information diffusion model as accurately as possible; (2) we would like to make sure the learned diffusion model is useful for influence maximization. Ultimately, we would like to make sure that the set of influential nodes found by using a greedy algorithm on a learned model is more influential than that found by the greedy algorithm on a randomly guessed model.

Motivated by these two objectives, in this paper we firstly empirically show that the influence maximization problem is sensitive to model parameters under the LT model. We define the problem of active learning of the LT model for influence maximization. In the sequel, we propose a weighted sampling algorithm to solve this active learning problem. Extensive experiments are conducted to evaluate our proposed algorithm on five networks. Results show that the proposed algorithm outperforms pure random sampling under the linear threshold model.

The rest of the paper is organized as follows. Section 2 reviews the preliminaries, including the LT information diffusion model and the study of parameter sensitivity under the LT model. We also define the problem of finding model parameters as an active learning problem in this section. Section 3 details our weighted sampling algorithm to learn the model parameters. Experimental results are shown in section 4. Finally, we review related work in section 5 and conclude in section 6.

2 Preliminaries and Motivation

In this section, we first introduce the LT information diffusion model. We then detail our study on the sensitivity of model parameters for the LT model, which motivates the necessity to learn the model parameters when they are unknown. We then formally define the problem of finding the model parameters for influence maximization as an active learning problem.

2.1 The Linear Threshold Model

In [9], because the threshold of each node is unknown, the influence maximization problem is defined as finding the set of nodes that can activate the largest expected number of nodes under all possible thresholds distributions. However, in our research, we assume that each nodes n_i in the network has a fixed threshold θ_i, which we intend to learn.

The *linear threshold model* [8, 9] assumes that one node u will be activated if the fraction of its activated neighbors are larger than a certain threshold θ_u. In a more general case, each neighbor v may have a different weight $w(u, v)$ to node u's decision. In this case, a node u becomes active if the sum of the weights of its activated neighbors is greater than θ_u. The requirement of a node u to become active can be described by the following equation:

$$\Sigma_v w(u, v) \geq \theta_u;$$

where v is an activated neighbor of u. In our research, we focus on finding the set of θ_us under the LT model.

For the convenience of presentation, we define the influence spread of a given set of nodes $|S|$ as the number of the nodes that are activated by the set S when the diffusion process terminates. In this paper we focus on the influence maximization problem on the LT model with fixed static unknown thresholds.

2.2 Sensitivity of Model Parameters

In this section we will empirically check whether the influence maximization problem is sensitive to the model parameters under the LT model.

To check the sensitivity of model parameters, we assume that there is a true model with the true parameters. We also have an estimated model. We use a greedy algorithm on the *estimated* model and get a set of influential nodes. Denote this set as $S_{estimate}$. We perform the greedy algorithm on the *true* model, get another set of influential nodes,

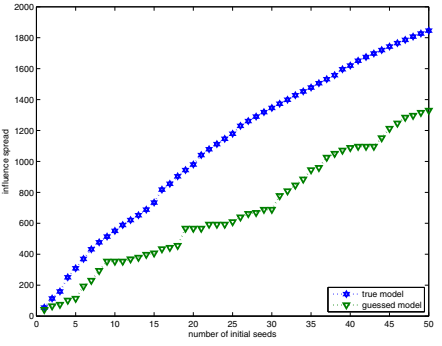


Fig. 1. Model Parameter Sensitivity Test on the GEOM network

and denote this set as S_{true}. We check the influence spread of $S_{estimate}$ and S_{true} on the true model. The sensitivity of models is then defined as follows: if the difference between S_{true} and $S_{estimate}$ is smaller than a given small number, we can infer that the influence maximization problem is not very sensitive to model parameters; otherwise it is sensitive to model parameters.

To test the sensitivity of the LT model, we assume that the thresholds of the *true* model are drawn from a truncated normal distribution with mean 0.5 and standard deviation of 0.25. Suppose all the thresholds in the *estimated* model are 0.5. Figure 1 shows that the influence spread of $S_{estimate}$ is significantly lower than that of the set S_{true}. We have conducted similar experiments on other collaboration networks and citation networks, and similar pattern can be found in those networks. These observations motivate us to find the parameters in the LT model for influence maximization.

2.3 Active Model Parameter Learning for Influence Maximization

Since the influence spread under the LT model is quite sensitive to model parameters, we now present a formal definition of active model parameter learning for the LT model for influence maximization. Notations used in our problem definition are presented in Table 1.

We assume that there is a true fixed threshold θ_i of each node n_i in the social network $G(V, E)$. Our goal is to learn θ as accurately as possible. In order to make the problem definition easier, we will actively construct the information diffusion data D over multiple iterations. In each iteration, we can use at most κ nodes to activate other nodes in the network. After we acquire the activation sequences in each iteration, the model parameters can be inferred. More specifically, we can infer the lower bound θ_{lowbd} and the upper bound θ_{upbd} of the thresholds of some nodes according to the activation sequences D. The details of the inference will be introduced in Section 3. With more and more iterations, we can get the thresholds θ more tightly bounded or even hit the accurate threshold value. The activation sequences of different iterations are assumed to be independent. That means at the beginning of each iteration, none of the nodes are activated (influenced). The above process can be summarized into the following three functions.

Table 1. Notations

Symbol	Meaning
$G(V, E)$	the social network
κ	the budget for learning in each iteration
θ	the true thresholds of all nodes
$\hat{\theta}$	the estimated thresholds of all nodes
D	the activation sequences
$M(\boldsymbol{\theta})$	the true Model
$M(\hat{\theta})$	the estimated Model
S_{true}	the set of influential nodes found by using the true model $M(\boldsymbol{\theta})$
$S_{estimate}$	the set of influential nodes found by using the estimated model $M(\hat{\theta})$
$f(S_{true}, M(\boldsymbol{\theta}))$	the influence spread of the set S_{true} on $M(\boldsymbol{\theta})$
$f(S_{estimate}, M(\boldsymbol{\theta}))$	the influence spread of the set $S_{estimate}$ on $M(\boldsymbol{\theta})$
$f(S_{estimate}, M(\hat{\theta}))$	the influence spread of the set $S_{estimate}$ on $M(\hat{\theta})$

$$f_1 : (G, \hat{\theta}, \theta_{lowbd}, \theta_{upbd}) \mapsto S \quad s.t. |S| = \kappa \tag{1}$$

$$f_2 : (G, M(\theta), S) \mapsto D \tag{2}$$

$$f_3 : (G, D, \hat{\theta}, \theta_{lowbd}, \theta_{upbd}) \mapsto \{\hat{\theta}', \theta'_{lowbd}, \theta'_{upbd}\} \tag{3}$$

Function (1) is the process of finding which set of nodes to target in each iteration. Function (2) is the process of acquiring the activation sequences D. Function (3) is the process of threshold inference based on the activation sequences and the old threshold estimate. In each iteration these three functions are performed in sequence.

In this setting, there are two questions to ask: (1) How to select the set S in each iteration so that the parameters learned are the most accurate; (2) When will the learned model parameters $\hat{\theta}$ be good enough so that it is useful for the purpose of influence maximization? More specifically, when will the influential nodes found on the estimated model provide a significantly higher influence spread than that found on a randomly guessed model. Our solution is guided by these two questions (or interchangeably, objectives). However it is difficult to combine these two questions into one objective function. Since our final goal is to conduct influence maximization, we rephrase these two objectives in the context of influence maximization as follows.

The first goal is that the influence spread of a set of nodes on the estimated model is close to the influence spread of the same set of nodes on the true model. This goal is in essence a prediction error. If they are close, it implies that the two models are close. The second goal is that the influential nodes found by using the estimated model will give

an influence spread very close to that found by using the true model. The second goal measures the quality of the estimated model in the context of influence maximization. We combine these two goals in the following equation.

$$\begin{aligned} Minimize|f(S_{true}, M(\boldsymbol{\theta})) - f(S_{estimate}, M(\boldsymbol{\theta}))| + \\ |f(S_{estimate}, M(\boldsymbol{\theta})) - f(S_{estimate}, M(\hat{\theta}))| \\ s.t. iterations = t. \end{aligned} \tag{4}$$

$|f(S_{true}, M(\boldsymbol{\theta})) - f(S_{estimate}, M(\boldsymbol{\theta}))|$ measures whether the set of influential nodes determined by using the estimated model can give an influence spread close to the influence spread of the set of influential nodes determined by using the true model. $|f(S_{estimate}, M(\boldsymbol{\theta}))-f(S_{estimate}, M(\hat{\theta}))|$ measures the difference of influence spreads between the true model and the estimated model. It is an approximation to measure the "model distance", which we will define in section 4.

3 The Weighted Sampling Algorithm

In this section we will firstly show the difficulty of the active learning problem and then present our algorithmic solution: the Weighted Sampling algorithm.

The difficulty of the above active learning problem is two-fold. The first difficulty is that even learning the exact threshold of a single node is quite expensive if the edges of a network are weighted.

Assume for each edge E in a social network $G(V, E)$, there is an associated weight w. For the simplicity of analysis, we assume $\forall w, w \in Z^+$. An edge $e = \{u, v\}$ is active if either u or v is active. What we can observe from the diffusion process is then a sequence of node activations (n_i, t_i). In this setting, suppose that at time t the sum of weights of the active edges of an inactive node n_i is c_t. At some future time t_k, the node n_i becomes active and the sum of the weight of the active edge at time is c_{t_k}. We use w_i to denote the sum of weights of all edges that connect to node n_i. We can infer that the threshold of node $n_i \in [c_t/w_i\ c_{t_k}/w_i]$. More specially if $c_{t_k} = c_t + 1$, the threshold is exactly c_{t_k}/w_i, and if this is the case, a binary search method can be used to determine the threshold of a node n_i deterministically, which is detailed as follows.

Assume that the set of edges connected to a node n_i is E_i. There is a weight w associated with each edge in E_i. S is the set of weights associated with E_i. Because $w \in Z^+$, this means that S is a set of integers. There is a response function $F : T \mapsto \{0, 1\}$ based on a threshold θ, where $T \subseteq S$. Here $\theta' = \theta * \sum(S)$, and $\sum(S)$ means the sum of elements of set S.

$$F(T) = \begin{cases} 1 & \sum(T) \geq \theta' \\ 0 & \sum(T) < \theta' \end{cases} \tag{5}$$

Given this response function, we can define θ' in equation (6).

$$\begin{aligned} \theta' = min(\sum(T)) \\ s.t. \quad F(T) = 1 \end{aligned} \tag{6}$$

Therefore the actual threshold θ is defined in equation (7).

$$\theta = min(\sum(T))/\sum(S) \quad s.t. \quad F(T) = 1; \tag{7}$$

Now we analyze the time and space complexity of this deterministic binary search. Assume that the set of possible values of $\sum(T)$ is $\mathbb{T}$. To find θ', we can sort $\mathbb{T}$ firstly and perform a binary search on the sorted list of $\mathbb{T}$. The time complexity is $O(log|\mathbb{T}|)$. $|\mathbb{T}|$ is $O(2^{|S|})$. So the time complexity of binary search is $O(log(|\mathbb{T}|)) = O(log(2^{|S|})) = O(|S|)$. However, sorting the set $\mathbb{T}$ will take $O(|\mathbb{T}|log|\mathbb{T}|)$ steps. So the overall time complexity is $O(2^{|S|} * |S|)$. In addition, the space requirement is $O(2^{|S|})$. In short, a deterministic binary search algorithm to learn the threshold of just one node is expensive. It will be infeasible to extend this approach to a large scale network with a large number of nodes.

Next we will introduce the second difficulty. This difficulty comes from the perspective of active learning algorithm design. We define the following function.

$$\Gamma : (S, G, M(\hat{\theta}), M(\theta)) \mapsto E(Red) \tag{8}$$

This function maps the target set S, the graph G, the estimated model $M(\hat{\theta})$ and the true model $M(\theta)$ to the expected reduction in threshold uncertainty $E(Red)$ if we set S as the initial active nodes. $\Gamma(S, G, M(\hat{\theta}), M(\theta))$ measures the gain if we select S as the initial target nodes. Since we do not know the true model parameters and therefore we cannot possibly know the activation sequence of a target set S under the true model parameters. It is therefore impossible to know the exact value of $\Gamma(S, G, M(\hat{\theta}), M(\theta))$. $\Gamma(S, G, M(\hat{\theta}), M(\theta))$ is not a monotonically non-decreasing function with respect to set S, which means even if we know the value of $\Gamma(S, G, M(\hat{\theta}), M(\theta))$, a deterministic greedy algorithm is not a good solution. However, we still want to choose the set S that maximizes $\Gamma(S, G, M(\hat{\theta}), M(\theta))$ in each learning iteration. We use weighted sampling to approximate this goal. In each iteration we sample a set of κ nodes according to the following three probabilities.

$$p_i \propto \sum_j \mathbf{I}(i, j) \tag{9}$$

$$p_i \propto \sum_j \mathbf{I}(i, j) * w(i, j) \tag{10}$$

$$p_i \propto \sum_j \mathbf{I}(i, j) * (\theta(j)_{upbd} - \theta(j)_{lowbd}) \tag{11}$$

$\mathbf{I}(i, j)$ is the indicator function. It is equal to 1 if there is an edge between i and j and the threshold of node j is unknown, otherwise it is 0. Essentially we are trying to sample κ nodes that connect to the most number of nodes with the most uncertainty of thresholds. There are different ways to measure the uncertainty of the threshold of a node. Formula

(11) measures the uncertainty by how tight the bounds of the threshold are. In formula (9) the uncertainty value is 1 if the threshold is unknown and 0 otherwise. Formula (10) differs from formula 9 in that weights of edges $w(i,j)$ are added. We perform weighted sampling on the nodes without replacement. The hope is that the sampled set S can yield a high value of $\Gamma(S, G, M(\hat{\theta}), M(\theta))$. The pseudo code of our weighted sampling algorithm is summarized in Algorithm 1.

Algorithm 1. Active Learning based on Weighted Sampling

Input: A social network G, the budget κ of each iteration, and the number of learning iterations t.
Output: The estimated threshold $\hat{\theta}$
Method:

1: Let p_i=the number of nodes with unknown thresholds that node i connects to.
2: normalize $\boldsymbol{p}$ to 1.
3: set $\hat{\theta}$ to be 0.5 for all nodes.
4: set θ_{lowbd} to be 1 for all nodes.
5: set θ_{upbd} to be the number of edges that each node connects to.
6: **for** $i = 1$ to t **do**
7: set all nodes in G to be inactive.
8: let S=sample κ nodes according to p without replacement.
9: set S as the initial nodes and start the diffusion process on the true model $M(\theta)$.
10: let (n,t) be the observed activation sequence.
11: update $\hat{\theta}$, θ_{lowbd}, and θ_{upbd} according to the activation sequence (n_i, t_i).
12: update the sampling probability $\boldsymbol{p}$ and normalize $\boldsymbol{p}$.
13: **end for**
14: return $\hat{\theta}$

Steps 5 to 13 show the learning process. We sample κ nodes in each iteration according to the sampling probability $\boldsymbol{p}$. We then set the κ nodes as the initial active nodes and simulate the diffusion process on the true model $M(\theta)$. After that we can observe a series of activation sequences (n_i, t_i). We can update the threshold estimation $\hat{\theta}$, θ_{lowbd} and θ_{upbd} accordingly. After that we update the sampling probability and the iteration ends.

4 Experimental Evaluation

In this section, we present the network datasets, the experimental setup and experimental results.

4.1 Network Datasets

Table 2 lists the datasets that we use. NetHEPT is from [3]. NETSCI, GEOM, LEDERB and ZEWAIL are from Pajek's network collections.

Table 2. Network datasets

Dataset	Name	n(# nodes)	m(# edges)
NETHEPT	the High Energy Physics Theory Collaboration network	15233	58891
GEOM	the Computational Geometry Collaboration network	7343	11898
NETSCI	the Network Science Collaboration network	1589	2742
ZEWAIL	the Zewail Citation network	6752	54253
LEDERB	the Lederberg Citation network	8843	41609

4.2 Experimental Setting

Even though [13] deals with a very similar problem to the problem in this paper, the methods in [13] is not directly usable in our problem setting. [13] assumes the diffusion data is available and the diffusion model is the IC model. In this paper we actively select seeds to obtain the diffusion data and we focus on the LT model. Methods in [13] can be used after we obtained the diffusion data. However that is not the focus here. Therefore we do not compare our methods to methods in [13].

For the simplicity of experiments, we treat all the networks as undirected. In the collaboration networks, the weight of an edge is set to be the number of times that two authors collaborated. For the citation network, the weights of all edges are randomly chosen from 1 up to 5. In the experiments we draw the thresholds of all nodes from a truncated normal distribution with mean 0.5 and standard deviation of 0.25, where all the thresholds are between 0 and 1.0. The number of iterations of learning is set to be 3000. The budget for each iteration is set to be 50, which means in each iteration 50 nodes are set as the initial seeds. In order to evaluate the objective function, the budget for influence maximization is also set to be 50 nodes.

Results of two versions of the weighted sampling are reported. The first version, denoted as $mctut$, uses the weighting scheme in formula (9). The second version, denoted as $metut$, uses the weighting scheme in formula (10). Experiments were also conducted on the weighting scheme according to formula (11). The results were slightly worse than the former two weighting schemes. The reason why the weighting scheme according to formula (11) is worse than (9) and (10) is because formula (11) biases towards learning the thresholds of the high degree nodes. The thresholds of the high degree nodes are more difficult to learn. It costs more iterations to infer the thresholds of the high degree nodes. However, in the distance calculation, the high degree nodes are treated the same as the low degree nodes. Therefore given the same iterations, the weighting schemes by formulas (9) and (10) would infer more thresholds than the weighting scheme by formula (11). In addition to these two schemes, we use un-weighted random sampling as a baseline, and denote it as $random$.

4.3 Experimental Results

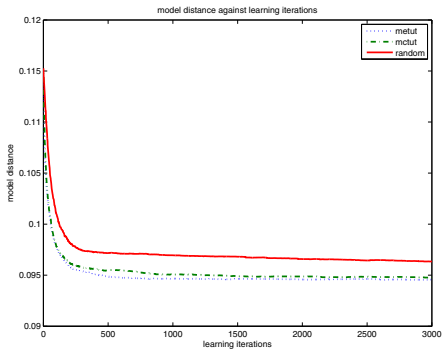
To evaluate the effectiveness of our algorithm, we use two performance metrics: *the objective function* (equation 4) and *model accuracy*. First we define *model distance* as the average threshold difference between the true model and the estimated model. *Model accuracy* is basically the inverse of *model distance*. So if the *model distance* is low, *model accuracy* is high and vice versa.

Figures 2(a),2(b),2(c),2(d),and 2(e) show that both $metut$ and $mctut$ are better than $random$ on all the networks in terms of model accuracy. There is little difference between $metut$ and $mctut$ in all these figures. The difference between $metut$ and $random$ is more noticeable in Figures 2(a),2(d) and 2(e). In Figure 2(b), the difference become noticeable when the number of learning iteration approaches 3000. In Figure 2(c) the difference is the largest between iterations 200 and 1000. After that the difference between $metut$ and $random$ decreases over iterations. The difference becomes quite small when the number of learning iterations approaches 3000. To sum up, we can see that both $metut$ and $mctut$ perform well. They are better than the baseline method and the model distance decreases with iterations.

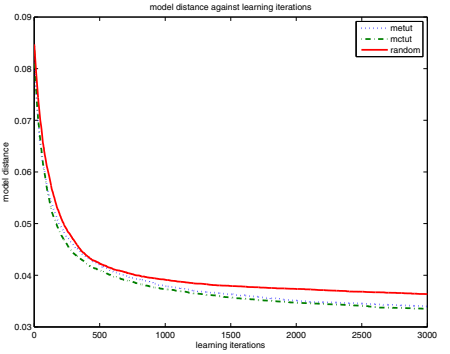
Figures 3(a),3(b),3(c),3(d) and 3(e) show that $metut$ beats $random$ on all the networks in terms of objective function. $mctut$ outperforms $random$ in Figures 3(a), 3(b),3(c) and 3(d). In Figure 3(e), $mctut$ is worse than $random$ at iteration 500. But after that $mctut$ is better $random$. $metut$ is more stable than $mctut$ in a sense. The absolute value of difference of the objective function is actually very large. In Figure 3(d) we can see that the largest difference between $metut$ and $random$ is about 500. So far we can assume that $metut$ is the best of the three in terms of objective function and model distance.

Finally we devise a way to measure the quality of the estimated model. If the influence spread of the solution found on the estimated model is very close to that found on the true model, we can say the estimated model has good quality, otherwise the estimated model has low quality. Figure 4(a) shows the quality of the estimated model found by $metut$ in learning iterations of 0 and 3000. We can see that the initial guessed model at iteration 0 has extremely low quality. The influence spread of influential nodes obtained by using the greedy algorithm on the estimated model at iteration 0 is noticeably less than that obtained by using the greedy algorithm on the true model. However after 3000 iterations of learning, this gap is narrowed sharply. The influence spread of influential nodes obtained by using the greedy algorithm on the estimated model at iteration 3000 is very close to that obtained by the using the greedy algorithm on the true model. This shows that the proposed algorithm can indeed narrow the gap between the influence spread of the influential nodes on the estimated model and the true model. Figures 4(b) and 4(c) show the quality of the estimated model found by $mutut$ and $random$ in learning iterations 0 and 3000.

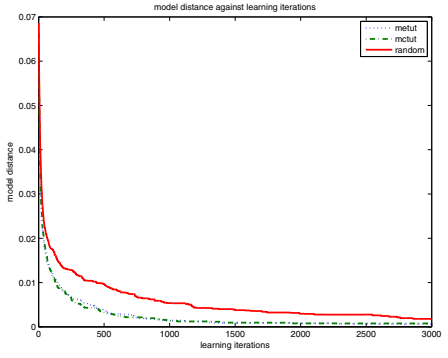
From Figures 4(a), 4(b) and 4(c) we can also observe that the estimated models found by $metut$ and $mutut$ have higher quality than the estimated model found by $random$ at iteration 3000. We can notice that the influence spread of the solution found by using the estimated models of $metut$ and $mutut$ is larger than the influence spread of the solution using the estimated model of $random$ at iteration 3000. It indirectly shows that $metut$ and $mutut$ learn models more accurately than $random$. Similar patterns can be observed in Figures 4(d), 4(e), 4(f), 4(j), 5(a), 5(b), 5(c), 5(d) and 5(e). Although the differences between $metut$, $mutut$ and $random$ are not as obvious as the case of Figures 4(a), 4(b) and 4(c). Figures 4(g), 4(h) and 4(i) are exceptions. Figures 4(g), 4(h) and 4(i) show that the learned models of $random$, $metut$ and $mctut$ have almost identical quality. This is probably because the dataset NETSCI is a small dataset.



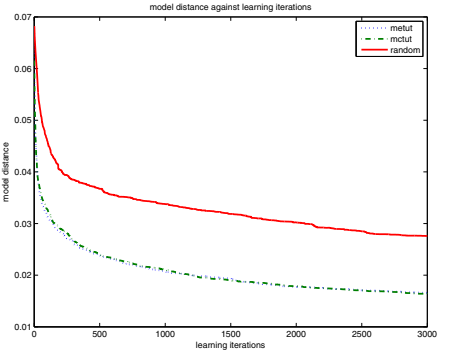
(a) GEOM



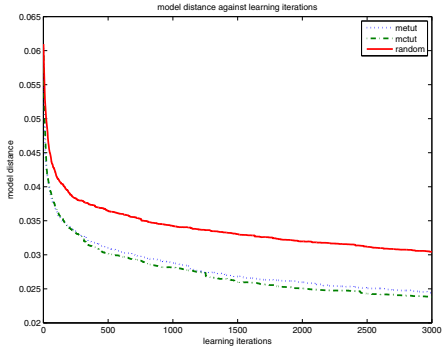
(b) NetHEPT



(c) NETSCI



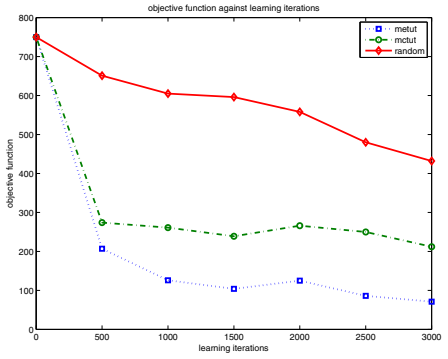
(d) LEDERB



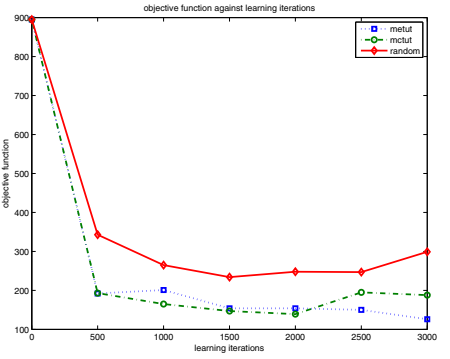
(e) ZEWAIL

Fig. 2. Comparison of metut,mctut and random in terms of model distance in different datasets (the lower the better)

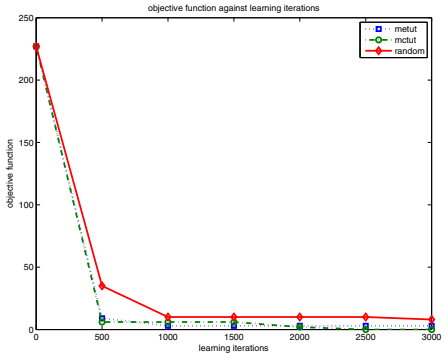
To this end, we have shown two points. Firstly, the learning process can indeed help narrow the gap between an estimated model and the true model. Secondly, $metut$ performs better than $random$ because of the fact that the solution found by using $metut$'s estimated model produces influence spread larger than the influence spread of the solution found by using $random$'s estimated model.



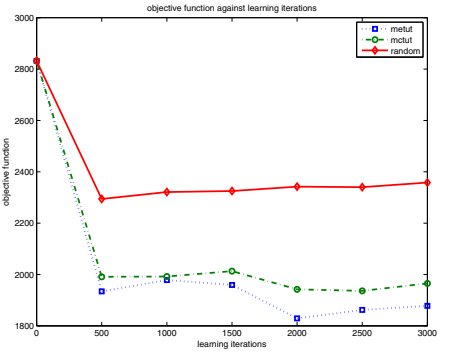
(a) GEOM



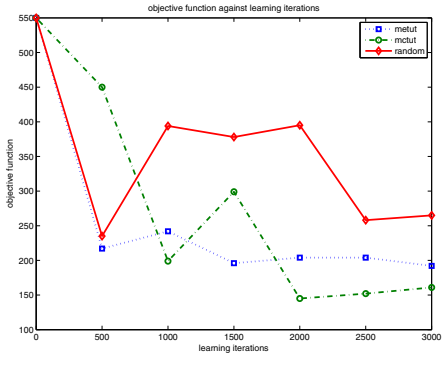
(b) NetHEPT



(c) NETSCI



(d) LEDERB

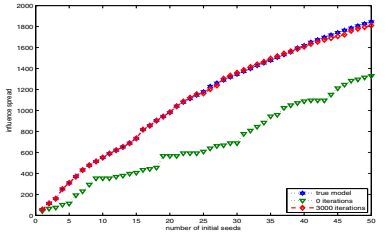


(e) ZEWAIL

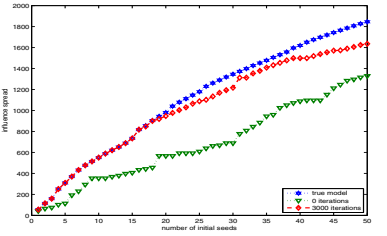
Fig. 3. Comparison of metut,mctut and random in terms of objective function in different datasets (the lower the better)

5 Related Work

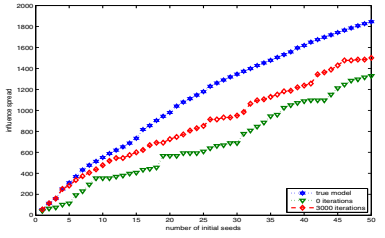
We review related works on influence maximization and learning information diffusion models in this section.



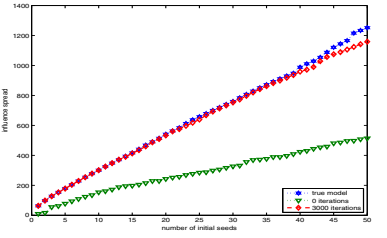
(a) metut on GEOM



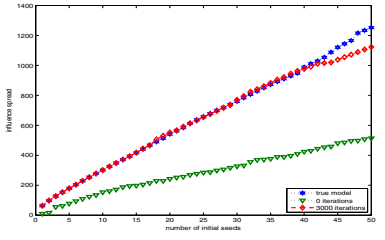
(b) mutut on GEOM



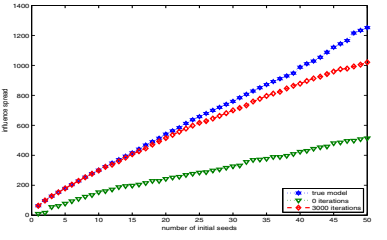
(c) random on GEOM



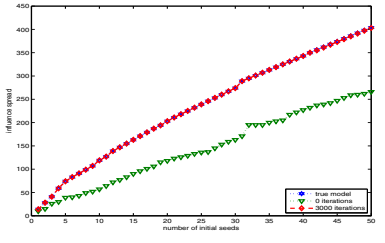
(d) metut on NetHEPT



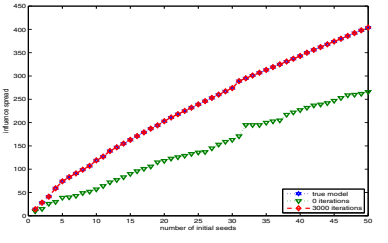
(e) mutut on NetHEPT



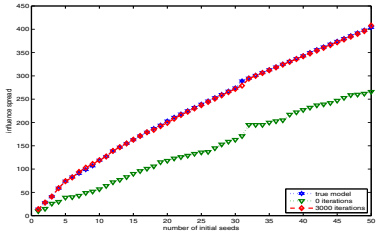
(f) random on NetHEPT



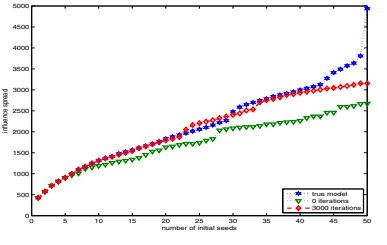
(g) metut on NETSCI



(h) mutut on NETSCI

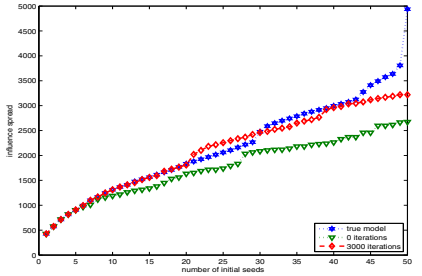


(i) random on NETSCI

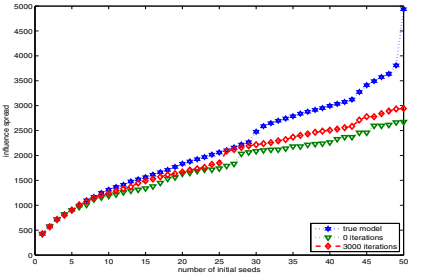


(j) metut on LEDERB

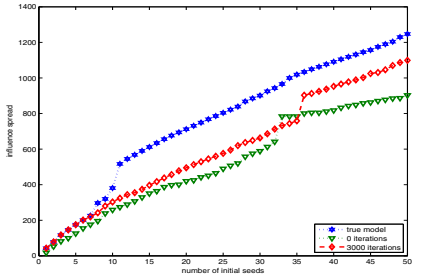
Fig. 4. Comparison of metut, metut and random on the improvement of influence spread over a guessed model on different networks



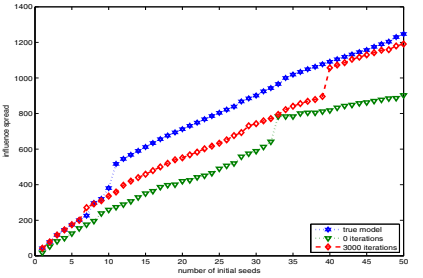
(a) mutut on LEDERB



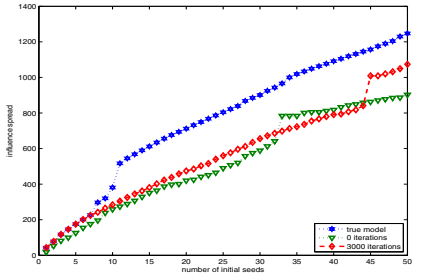
(b) random on LEDERB



(c) metut on ZEWAIL



(d) mutut on ZEWAIL



(e) random on ZEWAIL

Fig. 5. Comparison of metut, metut and random on the improvement of influence spread over a guessed model on different networks

First we review works on influence maximization. [9] proved that the influence maximization problem is an NP-hard problem under both the IC model and the LT model. [9] defined an influence function $f(A)$, which maps a set of active nodes A to the expected number of nodes that A activates. They proved that $f(A)$ is submodular under both models. Based on the submodularity of the function $f(A)$, they proposed a greedy algorithm which iteratively selects a node that gives the maximal margin on the function $f(A)$. The greedy algorithm gives a good approximation to the optimal solution in terms of diffusion size. Follow-up works mostly focus on either improving the running time of the greedy algorithm [10, 3] or providing faster heuristics that can give influence spread that is close to the greedy algorithm [3, 2, 4, 1, 14]. [3] used a Cost-Effective Lazy Forward method to optimize the greedy algorithm to save some time. [3] proposed

degree discount heuristics that reduce computational time significantly. [4] and [2] proposed heuristics based on a most likely propagation path. The heuristics in these two papers are tunable with respect to influence spread and computational time. [1, 14] used a community structure to partition the network into different communities, find the influential nodes in each community, and combine them together. In [9], the problem of calculating the influence function $f(A)$ was left as an open problem. Later [2] and [4] proved that it is ♯P hard to calculate the influence function $f(A)$ under both the IC and models. [10] performed the bound percolation process on graphs and used strongly connected component decomposition to save some time on the calculation of the influence function $f(A)$. All these works assume that the model parameters are given.

Then we review research efforts on learning information diffusion models. [7] proposed both static and time dependent models for capturing influence from propagation logs. [13] used the EM algorithm to learn the diffusion probabilities of IC model. [12] proposed asynchronous time delayed IC and LT models. After that they used maximum likelihood estimation to learn the models and evaluated the models on real world blog networks. Interestingly, [6] focused on a different problem, i.e., learning the network topology which information diffusion relies on. [6] used maximum likelihood estimation to approximate the most probable network topology.

6 Conclusions

In this paper, we have studied the influence maximization problem under unknown model parameters, specifically, under the linear threshold model. To this end, we first showed that the influence maximization problem is sensitive to model parameters under the LT model. Then we defined the problem of finding the model parameters as an active learning problem for influence maximization. We showed that a deterministic algorithm is costly for model parameter learning. We then proposed a weighted sampling algorithm to solve the active learning problem. We conducted experiments on five datasets and compared the weighted sampling algorithm with a naive solution: pure random sampling. Experimental results showed that the weighted sampling achieves better results than the naive method in terms of both the objective function and model accuracy we defined. Finally we showed that by using the learned model parameters from the weighted sampling algorithm, we can find the influential nodes that give an influence spread very close to the influence spread of influential nodes found on the true model, which further justifies the effectiveness of our proposed approach. In the future, we will investigate on how to learn the model parameters under the IC model.

Acknowledgments. The research is supported by the US National Science Foundation (NSF) under grants CCF-0905337 and NSF CCF 0905291.

References

[1] Cao, T., Wu, X., Wang, S., Hu, X.: Oasnet: an optimal allocation approach to influence maximization in modular social networks. In: SAC, pp. 1088–1094 (2010)
[2] Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: KDD, pp. 1029–1038 (2010)

[3] Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: KDD, pp. 199–208 (2009)
[4] Chen, W., Yuan, Y., Zhang, L.: Scalable influence maximization in social networks under the linear threshold model. In: ICDM, pp. 88–97 (2010)
[5] Domingos, P., Richardson, M.: Mining the network value of customers. In: KDD, pp. 57–66 (2001)
[6] Gomez-Rodriguez, M., Leskovec, J., Krause, A.: Inferring networks of diffusion and influence. In: KDD, pp. 1019–1028 (2010)
[7] Goyal, A., Bonchi, F., Lakshmanan, L.V.S.: Learning influence probabilities in social networks. In: WSDM, pp. 241–250 (2010)
[8] Granovetter, M.: Threshold models of collective behavior. The American Journal of Sociology 83(6), 1420–1443 (1978)
[9] Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. In: KDD, pp. 137–146 (2003)
[10] Kimura, M., Saito, K., Nakano, R.: Extracting influential nodes for information diffusion on a social network. In: AAAI, pp. 1371–1376 (2007)
[11] Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: KDD, pp. 61–70 (2002)
[12] Saito, K., Kimura, M., Ohara, K., Motoda, H.: Selecting information diffusion models over social networks for behavioral analysis. In: ECML/PKDD (3), pp. 180–195 (2010)
[13] Saito, K., Nakano, R., Kimura, M.: Prediction of information diffusion probabilities for independent cascade model. In: KES (3), pp. 67–75 (2008)
[14] Wang, Y., Cong, G., Song, G., Xie, K.: Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In: KDD, pp. 1039–1048 (2010)