University of British Columbia
Department of Computer Science

Learning Influence Diffusion Probabilities under the Independent Cascade with Independent Decay over Time

CPSC 534L Project Report

Group Members:
Saeid Naderiparizi - 98712169
Polina Zablotskaia - 97560163
Setareh Cohan - 98701162

January 5, 2018

CONTENTS

1 ABSTRACT

Influence maximization is a well-studied problem of finding a small set of highly influential individuals in a social network such that the spread of influence under a certain diffusion model is maximized. We first propose a diffusion model that incorporates the phenomenon that the power of influence decreases with elapsed time under the independent cascade model. We then learn the parameters of this model to get the influence probabilities. The motivation for doing this is that influence probabilities are usually not known for real-world social networks. We use two different datasets and two optimization algorithms which are Gradient Descent and Expectation Maximization. We evaluate the efficiency of our approach. In the end, we show the distribution error in time prediction .

2 INTRODUCTION

Recently, the rapidly increasing popularity of social networks has created opportunities to study models that simulate the spread of information in such networks. Examples of information spread in social networks are buying a product after a friend buys that product, sharing a post after a friend shares that post and retweeting a tweet of someone. In social networks, the weight or diffusion probability of an edge between user A to user B, is a measure of the influence from user A to user B. The higher the influence, the greater the value of the diffusion probability.

Probabilistic models such as Independent Cascade(IC) and the Linear Threshold(LT) have been used to study the spread of influence [7]. In the IC model, each edge is assigned with a diffusion probability and time increments in discrete steps. At time zero only a subset of nodes are active which are called the seed set. Each active neighbor v of u gets exactly one chance for activating u and succeeds with a probability of P_{vu} corresponding to edge vu. The active neighbors of vertex u can try to activate u in any order. When no new nodes can be activated, the diffusion process ends.
In the LT model, each edge uv of the graph is assigned to a value b_{uv} which captures the extent of influence on v from u. There exists a constraint that sum of weights of incomings edges to each node is not greater than one, $\sum_{\{u|u\rightarrow v\in E\}} b_{vu} \leq 1$. All nodes u randomly pick an activity threshold $\theta_u \in [0,1]$. Just like the IC model, time increments in discrete steps. At time zero we have only a subset of nodes active called the seed set. At time $t > 0$, node u gets active if the sum of weights of its incoming edges is greater than its threshold. When no new nodes can be activated, the diffusion process ends.

The power of influence depends on the elapsed time intuitively. For instance, if someone posts a funny picture on a website her friends are more likely to repost it the same week rather than three month later. We can argue that power of influence decreases over time. This motivated us to define probabilities which decay exponentially over time.

After we designed a time-decaying model based on the IC model, we proved that the influence spread function under this model is monotone and submodular. As shown in [8], we know that optimal solutions on monotone and submoldular functions can be approximated efficiently by a greedy solution.

Hence, we started working on a greedy algorithm that approximates the optimal solution to the influence maximization under our model. During this phase, we found a work by Ohsaka et al. in which they have proposed a model called Time-varying IC(TVIC) model which is a general case of our model [9] and they provide a scalable greedy algorithm for this mode.
Realizing that our model was already fully studied, we decided to focus on learning the probabilities of our model. The motivation behind this decision is that we need diffusion probabilities of edges both in our model and in TVIC, however these probabilities are usually not available for real networks. In this project, we will learn the probabilities of our model which we call *Independent Cascade with Independent Decay over Time(ICIDT).*

3 Related Work

Kempe et al. [7] formulated the influence maximization problem as a discrete optimization problem. They showed that this problem is NP-hard for IC model and the expected number of activated vertices with respect to an initial vertex set (called the seed set) is a monotone and submodular function. This implied that the optimal solution to influence maximization problem can be efficiently approximated within a ration of $(1-\frac{1}{e}-\epsilon)$ by a greedy algorithm [8]. More effective and scalable approaches to influence maximization have been proposed by [3] and [4]. However, these methods do not consider the effect of time on influences.

After proposing ICIDT model and showing that the influence function under this model is monotone and submodular, we started to work on finding an efficient greedy algorithm that well approximates the optimal solution. Meanwhile, we figured there are various diffusion models concerned with time. Although most of these consider the time difference between two vertices known as the time-delay [2][10][6], some works discuss the effect of elapsed time on influences [9][6].

The time-varying IC model (TVIC) proposed by Ohsaka et al. [9] formulates both time-delay and time-decaying phenomenon. In section 4.4 we show that our model is a special case of TVIC.

As mentioned earlier, IC model, TVIC model and our model(ICIDT) all require diffusion probabilities to be known in advance. This information may not be available for real networks, however, these probabilities can be estimated from a set of information diffusion cascades that are observed as time-sequences of activated nodes [6][11][1]. Saito et al. use the Expectation Maximization algorithm to predict diffusion probabilities [11]. Although the algorithm does reasonably well in predicting these probabilities, this approach is difficult to scale for large datasets. Goyal et al. propose static and time-dependent models that capture influence along with algorithms for learning the various parameters of the model they have presented [6]. However, they use a very specific formulation to model influence and learn parameters of this specific model. Hence, their approach is not general and cannot be directly applied to all other formulations of influence maximization problem. Furthermore, Saito et al. [12] address the problem of estimating diffusion probabilities of a probabilistic model as a function of node attributes from the observed diffusion data.

To our knowledge, there are no works on learning the diffusion probabilities of time-varying models. This motived us to propose a method for estimating the diffusion probabilities in the ICIDT model that we proposed.

4 Time-varying Model

In this section, we define ICIDT model and then we show that the influence function under our model is monotone and submodular.

4.1 Preliminaries

We extend the IC model [5][7]. In this model, for each directed link $e = (v, w)$, a real value p_{vw} is specified such that $0 < p_{vw} < 1$ in advance. Here, p_{vw} is referred to as the diffusion probability through link (v, w) The diffusion process proceeds from a given initial active seed set S as follows. When a node v becomes active at time-step t, it is given a single chance to activate each currently inactive child node w, and the attempt succeeds with probability p_{vw}. If multiple parents of a node become active at the same time $t*$, their activation attempts are sequenced in an arbitrary order, but all the attempts are performed at time-step $t*$. Whether v succeeds or not, it will not make any further attempts to activate w in the next time-steps. The process terminates if no more activations are possible.

4.2 Model Definition

Assume that we have an influence graph $G = (V, E)$ where V is the set of vertices and E is the set of edges of the graph. Each directed edge $(u, v) \in E$ is assigned a probability p_{uv} which we call the *initial influence probability*. Each node $v \in V$ is assigned a *user-specific decay factor*, λ_v. Assume node u was activated at time t_u. The probability of u influencing its neighbor v at time t, based on our model is

$$Pr[v \text{ getting activated at time } t | u \text{ is activated at time } t_u] = \begin{cases} p_{uv}e^{-\lambda_u(t-t_u)} & \text{if } t - t_u <= \Delta \\ 0 & \text{otherwise} \end{cases}$$

We can see that in our model, the probability of u getting activated by v goes to zero if the elapsed time is greater than a window size Δ.

Having graph G, we can define the influence function corresponding to the expected number of nodes that will be activated once the seed set S is activated in terms of possible worlds as follows:

$$\sigma(S) = \sum_{X \in PW} Pr[X]\sigma_X(S)$$

A possible world X is a graph (V, E') where $E' \subseteq E$ and each $(u, v) \in E'$ is assigned a *delay*, $d_{uv} \in \mathbb{N} \cup \{0\}$. For generating deterministic possible world graph from the probabilistic influence graph G, we toss at most Δ coins for each node such that the probability of getting a "head" in the k-th coin toss is $p_{uv}e^{-\lambda_u k}$ where $k \leq T$, and 0 otherwise. We set d_{uv} to the number of coin tosses until seeing the first "head" if it happens within our window T, and 0 otherwise.

4.3 Proof of Submodularity

In order to show that the influence function $\sigma(.)$ under ICIDT is monotone and submodular, we will look at the definition of this function in terms of possible worlds which was discussed earlier.

$\sigma_X(S)$ is the number of all reachable nodes from seed set S in possible world X, independent of the delays on the edges of X. Therefore, $\sigma_X(S)$ is the same as it was in the IC model and hence, we can conclude that $\sigma_X(S)$ is monotone and submodular just as in [7].
We know that $\sigma(S)$ is a linear combination of $\sigma_X(S)$ for different possible worlds with positive . This implies that $\sigma(S)$ is monotone and submodular.

4.4 Equivalency to TVIC

It can be shown that ICIDT model is a special case of TVIC model. In this model, probability of v becoming active at time t when u has been activated at time t_u is as below:

$$Pr[\text{v becoming active at time } t| \text{ u is active at time} t_u] = p_{uv}(t) f_{uv}(t - t_u) \tag{4.1}$$

Here, $f_{uv}(t - t_u)$ represents the time-delay effect on the edge $\{uv\}$ and $p_{uv}(t)$ is a non-increasing function on elapsed time.

Comparing the probability 4.1 with ICIDT, we can say that probability of v becoming active at time t when u has been activated at time t_u under ICIDT is equal to 4.1 when we only consider the non-increasing function on elapsed time and set the time-delay effect on edges to one.

At this point, we decided not to give up on our model and switch our focus to learn the diffusion probabilities on edges since we found no other works on this subject, and also these diffusion probabilities are usually not available for real-world networks.

5 Learning Probabilities under ICIDT

5.1 Preliminaries

For a given directed network or graph $G = (V, E)$ where V is the set of nodes or vertices and E is the set of links or edges. If u and v are different vertices of the graph and there exists an edge from u to v, we show this edge by (u, w). For each node v in the graph G, we define $F(v)$ as the set of child nodes of v as follows:

$$F(v) = w : (v, w) \in E.$$

Similarly, we define $B(v)$ to be the set of parent nodes of v as follows:

$$F(v) = u : (u, v) \in E.$$

In this section, we define the mathematical notation and formulate the problem of learning the parameters of our model. Looking back at our model, the probability of node w getting activated at time t_w by node v which is activated at time t_v is

$$p_{vw}e^{-\lambda_v(t_w-t_v)}$$

where p_{vw} is the initial influence probability and λ_v is the user specific decay factor.

In order to learn the edge probabilities, we need to learn its two parameters which are the initial influence probability of each edge and the user specific decay factor of each node. However, in order to make the problem simpler, we only learn the initial influence probabilities and assume that the decay factor is equal for all nodes and is a known value. Details on the value of decay factor is available in the Experiments section.

5.2 Problem Formulation

Here, we will formulate the problem of learning the initial influence probabilities and thus learning the edge probabilities.

Assume that we know the precise information for each past cascade, meaning which node was influenced at what time. The total number of these cascades is S. T_s represents the time when the diffusion process for cascade s ends. We represent the set of nodes which become active at time t under the cascade s as $D_s(t)$. The cumulative set of nodes which become active by the time t under cascade s are represented by $C_s(t)$. Therefore we can say that $C_s(t) = \bigcup_{t=0}^{t=T} D(t)$.

Let $P_w^s(t+1)$ represent the probability of w getting activated at time $t+1$ under the diffusion cascade s. In order to derive $P_w^s(t+1)$, we need to look back at our model and see what nodes are able to activate w. Based on our model, if a node is activated at some time t, it can activate its neighbors with a probability that decays over time and goes to zero after time Δ. It means that every node is contagious for its neighbors during a time window of size Δ. Using this information, we can formulate $P_w^s(t+1)$ as below:

$$P_w(t+1) = 1 - \prod_{\tau=0}^{\Delta-1} \prod_{v \in B(w) \cap D(t-\tau)} (1 - p_{vw}e^{-\tau/\eta}) \tag{5.1}$$

Now, we will define the likelihood of probabilities given the cascades. Assume v is a parent of w. In the case taht $v \in D(t)$ and $w \notin C(t+1)$, we know that the node v definitely failed in activating the node w through the link (vw). In the two cases below, we cannot obtain any information about the activation attempt through vw:

- $v \notin D(t)$.
- $v \in D(t)$ and $w \in C(t)$.

Therefore, for an episode D (which is basically the set of observed data) we can define the likelihood function with respect to $\theta = \{p_{vw}\}$:

$$L(\theta; D) = \prod_{t=0}^{T-1} \left[\left(\prod_{w \in D(t+1)} P_w(t+1) \right) \left(\prod_{\tau=0}^{\Delta-1} \prod_{v \in D(t-\tau)} \prod_{w \in F(v) \setminus C(t+1)} (1 - p_{vw}e^{-\tau/\eta}) \right) \right]$$

Let $\{D_s : s = 1, ..., S\}$ be the set of S different diffusion cascades. We can define log likelihood which will be our objective function with respect to θ:

$$L(\theta) = \sum_{s=1}^{S} \log(\theta, D_s) = \sum_{s=1}^{S} \sum_{t=0}^{T_s-1} \left[\sum_{w \in D_s(t+1)} \log P_w^{(s)} + \sum_{\tau=0}^{\Delta-1} \sum_{v \in D(t-\tau)} \sum_{w \in F(v) \setminus C(t+1)} \log(1 - p_{vw} e^{-\tau/\eta}) \right] \tag{5.2}$$

where $P_w(t+1)$ can be calculated using Eq.5.1.

Our problem now is finding the set of information diffusion probabilities, θ^*, that maximize the log likelihood in Eq.5.2:

$$\theta^* = \text{argmax}\, L(\theta).$$

5.3 Learning Methods

We use 2 different methods for finding the maximum likelihood estimate for the log likelihood formulated in Eq. 5.2. In this section, we describe each of the optimization strategies.

5.3.1 Gradient Descent

One of the most straight forward ways of maximizing a function is Gradient Descent. Here, $L(\theta)$ is the function we want to maximize with respect to θ. First, we initialize θ to some random estimates. Then, we iteratively find the gradient of the log likelihood function L for the current estimates of θ and move in the direction of the gradient with a certain step size α. Since we move in the direction of gradient at each iteration, we move in the direction in which the function increases the most. Therefore, we converge a local maxima. Formally, the iteration can be presented as below:

$$\theta^{i+1} = \theta^i - \alpha g(L(\theta^i)) \tag{5.3}$$

where i is the current gradient descent iteration, $g(L(\theta^i))$ is the gradient of the likelihood function for the current estimate of θ (θ^i) and α is the current step size. The equation below gives the expression for $g(L(\theta^i))$:

$$\frac{\partial L}{\partial p_{vw}} = \sum_{s=1}^{S} \begin{cases} \left[\dfrac{1}{P_w^{(s)}} \dfrac{\partial P_w^{(s)}}{\partial p_{vw}} + Z_{vw}^{(s)} \right] & \text{if } v \in B(w) \ \&\ t_v < t_w \\ 0 & \text{otherwise} \end{cases}$$

where $\frac{\partial P_w^{(s)}}{\partial p_{vw}}$ can be calculated as

$$\frac{\partial P_w^{(s)}}{\partial p_{vw}} = \begin{cases} \left(\prod_{\tau=0}^{\Delta-1} \prod_{\substack{v' \in B(w) \cap D(t_w-\tau-1) \\ v' \neq v}} (1 - p_{v'w} e^{-\tau/\eta}) \right) e^{-(t_w - t_v - 1)/\eta} & \text{if } t_v^{(s)} \in [t_w^{(s)} - \Delta - 1, t_w^{(s)} - 1] \\ 0 & \text{otherwise} \end{cases}$$

and Z_{vw} is equal to

$$Z^{(s)}_{vw} = -\sum_{\tau=0}^{\min\{t_w - t_v - 2, \Delta - 1\}} \frac{1}{1 - p_{vw} e^{-\tau/\eta}} e^{-\tau/\eta}$$

The size of step size can be constant over all the iterations or it can be decreased as the iterations continue.

5.3.2 Expectation Maximization

Another way of maximizing the likelihood is using Expectation Maximization method. This method is used by Saito et al. [11] to learn the diffusion probabilities under the IC model. Expectation Maximization is an iterative method for finding the Maximum Liklihood Estimate(MLE) of a likelihood function. This method iteratively applies the Expectation step(E step) and the Maximization step(M step). In the E step we need to caculate the Q-function which is the expected value of the log likelihood function, with respect to the conditional distribution of the latent variables and observations. Then, in the M step, we find the parameters that maximize the Q-function.

For our model, there are no explicit latent variables. Instead we treat the probabilities $P^s_w(t+1)$ as the latent variables and the parameters to be estimated are the initial influence probabilities, p_{vw}. The Expectation Maximization we use iteratively performs the below two steps until the Q-function converges, i.e. the change in the parameter estimates becomes less than a certain threshold.

- E step: Calculate the expected value of the log likelihood function $L(\theta)$, with respect to the conditional distribution of $P^s_w(t+1)$ under the current estimate of the parameters p_{vw}

$$Q(\theta|\theta^t) = E_{Z|X,\theta^t}[log(L(\theta; X, Z))] \tag{5.4}$$

  where Z is the altent variable in the equation above.

- M step: Update the value of estimate of the parameters p_{vw}

$$\theta^{(t+1)} = \underset{\theta}{\operatorname{argmax}}\, Q(\theta|\theta^t) \tag{5.5}$$

The Expectation Maximization algorithm is sensitive tot he initial estimate and is also prone to converging a local maxima instead of a global maxima. Just like with Gradient Descent, multiple random starting points and averaging over all outputs is used to alleviate these problems.

The Q-function in our problem is

$$\begin{aligned} Q(\theta|\hat{\theta}) = \sum_{s=1}^{S} \sum_{t=0}^{T_s - 1} \sum_{v \in D_s(t)} \sum_{\tau=0}^{\Delta - 1} \Bigg[& \sum_{w \in F(v) \cap D_s(t+1+\tau)} \left(\frac{\hat{p}_{vw} e^{-\tau/\eta}}{\hat{P}^{(s)}_w} \log(p_{vw} e^{-\tau/\eta}) + (1 - \frac{\hat{p}_{vw} e^{-\tau/\eta}}{\hat{P}^{(s)}_w}) \log(1 - p_{vw} e^{-\tau/\eta}) \right) \\ & + \sum_{w \in F(v) \setminus C_s(t+\tau+1)} \log(1 - p_{vw} e^{-\tau/\eta}) \Bigg] \end{aligned} \tag{5.6}$$

where $\hat{\theta}$, $\hat{p}_{vw}$ and $\hat{P}^s_w(t+1)$ represent the current estimates of θ, p_{vw} and $P^s_w(t+1)$ respectively.

Next, in order to perform the M step, we can set the value of the gradient of Q-function with respect to θ to zero. Let $S^+_{v,w}$ be the set of episodes s which satisfy both $v \in D_s(t)$ for some $t \in [0, T_s]$ and $w \in D_s(t+\tau+1), 0 \leq \tau < \Delta$ (which means $t^{(s)}_w - t^{(s)}_v \leq \Delta$). And let $S^-_{v,w}$ be the set of episodes s which satisfy both $v \in D_s(t)$ for some $t \in [0, T_s]$ and $w \in D_s(t+1+d)$ for some $d \geq \Delta$ (which means $t^{(s)}_w - t^{(s)}_v > \Delta$). Also, let $A(x) = e^{x/\eta}$:

$$\frac{\partial Q}{\partial p_{vw}} = 0 \tag{5.7}$$

$$\begin{aligned}
\Rightarrow \sum_{s \in S^+_{v,w}} & \left[\sum_{\tau=0}^{t^{(s)}_w - t^{(s)}_v - 1} \frac{\hat{p}_{vw} e^{-\tau/\eta}}{\hat{P}^{(s)}_w} * \frac{1}{p_{vw}} - \left(1 - \frac{\hat{p}_{vw} e^{-\tau/\eta}}{\hat{P}^{(s)}_w}\right) * \frac{e^{-\tau/\eta}}{1 - p_{vw} e^{-\tau/\eta}} \right] \\
- \sum_{s \in S^-_{v,w}} & \left[\sum_{\tau=0}^{t^{(s)}_w - t^{(s)}_v - 1} \frac{e^{-\tau/\eta}}{1 - p_{vw} e^{-\tau/\eta}} \right] = 0
\end{aligned} \tag{5.8}$$

$$\Rightarrow \sum_{s \in S^+_{v,w}} \left[\sum_{\tau=0}^{t^{(s)}_w - t^{(s)}_v - 1} \frac{\hat{p}_{vw} e^{-\tau/\eta}}{\hat{P}^{(s)}_w} * \left(\frac{1}{p_{vw}} + \frac{e^{-\tau/\eta}}{1 - p_{vw} e^{-\tau/\eta}} \right) \right] = \sum_{s \in S^-_{v,w}} \left[\sum_{\tau=0}^{t^{(s)}_w - t^{(s)}_v - 1} \frac{e^{-\tau/\eta}}{1 - p_{vw} e^{-\tau/\eta}} \right]$$

$$\Rightarrow \sum_{s \in S^+_{v,w}} \left[\sum_{\tau=0}^{t^{(s)}_w - t^{(s)}_v - 1} \frac{\hat{p}_{vw}}{\hat{P}^{(s)}_w * e^{\tau/\eta}} * \left(\frac{1}{p_{vw}} + \frac{1}{e^{\tau/\eta} - p_{vw}} \right) \right] = \sum_{s \in S^-_{v,w}} \left[\sum_{\tau=0}^{t^{(s)}_w - t^{(s)}_v - 1} \frac{1}{e^{\tau/\eta} - p_{vw}} \right]$$

$$\Rightarrow \sum_{s \in S^+_{v,w}} \left[\sum_{\tau=0}^{t^{(s)}_w - t^{(s)}_v - 1} \frac{\hat{p}_{vw}}{\hat{P}^{(s)}_w * e^{\tau/\eta}} * \frac{e^{\tau/\eta}}{p_{vw}(e^{\tau/\eta} - p_{vw})} \right] = \sum_{s \in S^-_{v,w}} \left[\sum_{\tau=0}^{t^{(s)}_w - t^{(s)}_v - 1} \frac{1}{e^{\tau/\eta} - p_{vw}} \right]$$

6 Experiments

In this section we describe the experimental setup used to benchmark our results. In particular, we describe each aspect of the setup - the dataset used, implementation details, the algorithms considered and the evaluation metrics used.

6.1 Dataset

We used two different datasets:

- Twitter with 1.7 million users and 308 million edges.
  - Reduced to 9000 users and approximately 33000 edges.
- Facebook with 4000 nodes and 88000 edges.
  - We used the structure of the network from the Facebook dataset and generated the corresponding graph. We needed to assign the initial influence probabilities to each edge since the dataset does not provide these values. We decided to randomly assign the diffusion probabilities in the range $(0, 1)$ for each edge.

6.2 Implementation Details

We have implemented gradient descent in two different programming languages namely Python and C++. Originally we have chosen Python due to it high feasibility in working with large datasets and the availability of various libraries for the data processing. This helped us a lot with the huge Twitter dataset, however we soon encountered the main Python drawback, such as very slow computational performance. Due to that reason we were unable to fairly estimate performance of our algorithm and had to switch to more efficient C++. After developing new algorithm implementation on C++, we had significant boost of computational time which eventually allowed us to partially estimate obtained results for the Facebook graph where we have simulated the spread.

6.3 Evaluation

In order to evaluate our work, we decided to perform two actions:

- First, we need to check the accuracy of our learning method. To do this, we real social network graph and generate the cascades by choosing edge probabilities randomly. Then, we learn the probabilities and compare them with the real, known ones.

  Even though having highly inaccurate probabilities does not necessarily mean that we cannot model social network's information cascade well, it is a good metric for knowing how we do on the learning side.

- We also evaluate the behavior of the cascades when they are simulated with the learned probabilities. To do so, we consider the cascades up to a particular time T as our training set and compare the cascades generated from the learned probabilities with the real cascades from time T onwards.

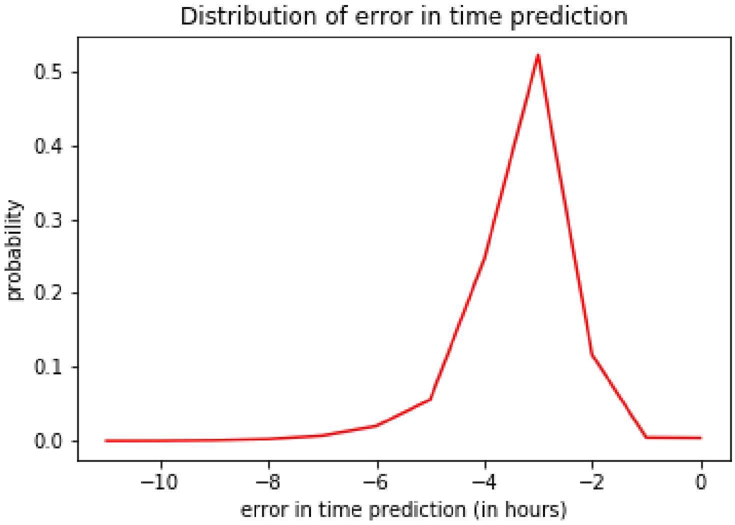
7 Results

We haven't been able to obtain all the desired data to evaluate our model, however for the simulated cascades data, we were able to compare the learned probabilities with randomly generated. To do so we decided to look at the predicted activation timestamps. On Figure 7.1 X-axis is the error in predicting time and Y-axis is the probability of the model making that error. Note that we observe an error measured in hours, which clearly shows that obtained results have quite high accuracy, even though model makes mistakes.

8 Conclusions

We proposed a diffusion model that captures effect of elapsed time on diffusion probabilities called ICIDT. We then showed that the diffusion function under this model is monotone and submodular. We then addressed the problem of inferring diffusion probabilities under this model. Particularly,

Figure 7.1: Error distribution



we planned to use two different optimization techniques - Gradient Descent and Expectation Maximization to learn these probabilities from past cascades. In the end, we only tried Gradient Descent because of lack of time. We evaluated our results by two different approaches on two different datasets.

9 Future Work

Currently, we learn the probabilities in ICIDT. Also, for simplicity, we only learn the initial diffusion probabilities. Next thing we can do is to learn the user specific decay factors along with the initial diffusion probabilities. We should note that learning user specific decay factor will introduce new challenges.
Our final goal could be to learn probabilities of TVIC since this model is a more general form of our model and also, there are no works on this model towards learning the diffusion probabilities as far as we know.

To make our current results better, we can do is to make our initial random guesses of initial probabilities smaller and make the number of cascades bigger. Also, we could try Stochastic Gradient Descent to converge faster. Moreover, we could implement the Expectation Maximization and evaluate results of this method. We could also try other optimization methods.

REFERENCES

[1] CAO, T., WU, X., HU, T. X., AND WANG, S. Active learning of model parameters for influence maximization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2011), Springer, pp. 280–295.

[2] CHEN, W., LU, W., AND ZHANG, N. Time-critical influence maximization in social networks with time-delayed diffusion process.

[3] CHEN, W., WANG, C., AND WANG, Y. 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* (2010), ACM, pp. 1029–1038.

[4] CHEN, Y.-C., PENG, W.-C., AND LEE, S.-Y. Efficient algorithms for influence maximization in social networks. *Knowledge and information systems 33*, 3 (2012), 577–601.

[5] GOLDENBERG, J., LIBAI, B., AND MULLER, E. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters 12*, 3 (2001), 211–223.

[6] GOYAL, A., BONCHI, F., AND LAKSHMANAN, L. V. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining* (2010), ACM, pp. 241–250.

[7] KEMPE, D., KLEINBERG, J., AND TARDOS, É. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (2003), ACM, pp. 137–146.

[8] NEMHAUSER, G. L., WOLSEY, L. A., AND FISHER, M. L. An analysis of approximations for maximizing submodular set functionsâĂŤi. *Mathematical Programming 14*, 1 (1978), 265–294.

[9] OHSAKA, N., YAMAGUCHI, Y., KAKIMURA, N., AND KAWARABAYASHI, K.-I. Maximizing time-decaying influence in social networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2016), Springer, pp. 132–147.

[10] RODRIGUEZ, M. G., BALDUZZI, D., AND SCHÖLKOPF, B. Uncovering the temporal dynamics of diffusion networks. *arXiv preprint arXiv:1105.0697* (2011).

[11] SAITO, K., NAKANO, R., AND KIMURA, M. Prediction of information diffusion probabilities for independent cascade model. In *Knowledge-based intelligent information and engineering systems* (2008), Springer, pp. 67–75.

[12] SAITO, K., OHARA, K., YAMAGISHI, Y., KIMURA, M., AND MOTODA, H. Learning diffusion probability based on node attributes in social networks. In *International Symposium on Methodologies for Intelligent Systems* (2011), Springer, pp. 153–162.