Prediction of Social Influence from Similarity-based and Context-aware Feature Learning

Abstract

Modelling the reposting behavior of users in microblog platforms is the key to sound understanding of dynamics in information cascades and therefore lays the foundation of real-world applications such as viral marketing. In social networks, traditional methods mainly model the influence from multiple in-neighbors by the sum of the pair-wise influence. Thus they suffer from high computational and space complexity ($O(m)$, where m is the number of edges) and the "overfitting" problem. In this project, we follow the LIS (latent influence and susceptibility) method [1] which mitigates these two problems and extends it with consideration of similarity between users' influence and susceptibility characteristics. Specifically, we propose a novel model for the probabilities of users' reposting behaviors based on the similarity between the influence and susceptibility vectors of the same user and/or the neighboring users. We also develop and analyze optimization methods for parameter estimation in our proposed model. We have conducted experiments on real-world information cascades and social networks from Sina Weibo and Twitter, which validates the performance improvement using our model compared to previous work.

1 Introduction

Thanks to the advent of social network services, nowadays researchers can observe human behaviors resulting from social influence and study the mechanism behind them by sample data crawled from API of online social network services like Twitter.

Retweeting, as one of the most frequently observed user behavior in microblog platforms, is important for understanding social influence in social networks. It consists of the classic two-stage process of social influence: 1). The user starts to be influenced as at least one neighbor of the user tweeted or retweeted some message. 2). The user makes the decision to retweet or not. Modeling social influence by retweeting behavior has been studied by researchers in previous works such as [3], [7], [8] and [9].

The first category of models, such as those proposed by [8] and [9], proposed data-driven models based on the famous independent cascade model proposed in [10]. They suffer from two problems: 1). A influence probability is needed for every edge in the network, making the space complexity $O(n^2)$ in worst case (n is the number of nodes). 2). The 'overfitting' problem, where the model is unable to predict unprecedented retweeting path. While the second category of models, like those from [3], are not theoretically justifiable as the features engineering are based on specific distribution from empirical analysis on a certain dataset. Therefore, the models might not be generalized to other datasets.

In this course project, as [1] proposed, we model the probability of retweeting for each user under certain social context by learning two latent vectors for representation of influence and susceptibility respectively. The term *social context* denotes the previous active neighbors of the user, they retweeted the message so that the user is under influence. This Latent Influence and Susceptibility (LIS) model introduced in [1] overcomes the overfitting problem mentioned before and only needs to store $O(2kn)$ parameters, where $k \ll n$ is the number of latent dimensions. We take one step further to take a novel factor into consideration: the influence and susceptibility of the same user should be similar. For example, Yann LeCun's original tweets about AI and deep learning are frequently retweeted by his followers while he also often does retweeting from messages related to

these two topics. Which means a user's influence and susceptibility should be similar. Following this intuition, we add the user-specific regularization term to the objective function of LIS model to control the direction of the two latent feature vectors for each user. For convenience, we name the proposed model as *LIS-similarity*. In the experiments, we evaluate the proposed model with two real-world datasets and compare the performance with the original LIS and the random guess.

the rest of this report is organized as follows: In Section 3, we introduce the problem formulation and the probabilistic model in detail: how inner product of latent vectors are used for calculation of retweeting probability. In Section 4, we show the final form of the convex optimization problem as a combination of negative log-likelihood and regularization terms. We also introduce the algorithm applied for solving this problem and justify this choice. In Section 5, we first describe the experimental setup and then show the dataset description and experimental results. Finally in Section 6, we make conclusion about the model, the algorithm and the experiments. We also include a brief description of future work of modeling retweeting probability by learning latent feature vectors.

2 Problem Formulation

The main focus of this project has been to predict whether a user will forward a message in the course of the lifecyle in which the message persists.

For a message m, a cascade for m is defined as a list of users activated in order of time denoted by $< a_1^m, a_2^m, \dots a_n^m >$, n being the number of reshares, where the term **active** means forwarding the message m.

Cascade context: A cascade context arises when an activated user a_i^m tries to activate an uninfected user v and the context for that attempt is defined as below:

$$\mathcal{D}_{v,i}^m = \{a_j^m | j \leq i, \delta(a_j^m, v) = 1\} \tag{1}$$

So for a particular cascade attempt on user v by an active user a_i, the list $\mathcal{D}_{v,i}^m$ denotes the previously activated users who are neighbors of v in the historical propagation network DN. So, essential there can be multiple cascade context attempts for a single user v in the course of the cascade lifecycle.

We model this prediction problem as a binary prediction problem in which the positive case occures when a user forwards a message over an attempt by an active user represnted by the corresponding cascade context and the negative case is the opposite situation when the user does not respond.

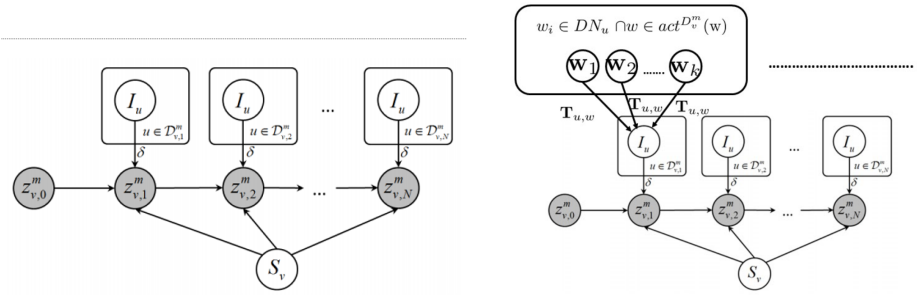
3 System Model

First we will lay down the baseline diffusion model used in [1] and then we will introduce the improvements we experimented through the course of this project.

Status vector: For every user v in the diffusion network DN, the status vector for m is defined as an n length vector $\mathbf{z}_v^m$ where the state $z_{v,j}^m$ indicates whether the user was activated right after exposure from user a_j^m. Each such state $z_{v,j}$ is a binary element where 0 indicates user a_j's attempt was a failure and vice versa if that state assumes a value 1.

Since we are interested in predicting whether the user v would forward message m, the problem boils down to modeling the status vector as follows:

$$P(\mathbf{z}_v^m | \delta) = p(z_{v,0}^m) \prod_{i=1}^{N} p(z_{v,i}^m | z_{v,i-1}^m, \mathcal{D}_{v,i}^m, \delta) \tag{2}$$



(a) Baseline Model (b) Our model with trust propagation

Figure 1: Graphical model representation for the probability models used in (a) [1] (b) Our model

where $z^m_{v,0}$ is introduced only for simplifying the notation. The first term denotes whether v is the source of message m, defined as

$$p(z^m_{v,0} = 1) = \begin{cases} 1, & v \text{ is the source} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The way the transition of user's status is modeled using the concept of first order Markov chain is as follows:

$$\begin{aligned} &p(z^m_{v,i} = 1 | z^m_{v,i-1} = 1, \mathcal{D}^m_{v,i}, \delta) = 1 \\ &p(z^m_{v,i} = 1 | z^m_{v,i-1} = 0, \mathcal{D}^m_{v,i}, \delta) = 1 \ - \ exp(-\lambda \delta(a^m_i, v) \sum_{u \in \mathcal{D}^m_{v,i}} I^T_u S_v) \\ &p(z^m_{v,i} = 0 | z^m_{v,i-1} = 0, \mathcal{D}^m_{v,i}, \delta) = 1 - p(z^m_{v,i} = 1 | z^m_{v,i-1} = 0, \mathcal{D}^m_{v,i}, \delta) \end{aligned} \tag{4}$$

The above model formulates the three possible cases of transition for a user v to state i under the Markov chain assumptions. The first states that the probability that a user remains activated given that it is activated in any of previous states is always 1, that is there is no going back to get deactivaed. The second statement models the case where a user gets activated in state i given that it was not activated yet in state $i-1$. Here the authors in [1] take into account the dot product of the latent feature vectors influence I and susceptibility S. The last formula denotes the case where a user remains inactive in the current state given that it has not been activated before.

Our model with Trust propagation: This kind of user influence has been widely applied in user item rating in recommendation systems where it has been widely argued [11] that user trust plays a very major role in influencing other users' ratings on an item. We capture the same idea through our model by including a model of trust propagation in the baseline model.

Due to social influence, we hypothesize that the behavior of a user u is affected by his direct neighbors $v \in N_{DN}$, the diffusion network. Let the number of times such a propagation has happened in the diffusion history DN be denoted by $C_{u,v}$. We denote the trust factor between two users u, v, $T_{u,v}$ as the trust that v has on u as follows:

$$T_{u,v} = \begin{cases} \frac{C_{u,v}}{\sum_w C_{w,v}}, & \text{if } w \in N_{DN}(v) \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $N_{DN}(u)$ denotes the neighbors of u in the diffusion network D.

In order to incorporate the effect of trust of a user's neighbors in the cascade contexts, we use the trust propagation model used in [11]. We formulate the transition status including the effect of trust among users as follows:

$$p(z^m_{v,i} = 1 | z^m_{v,i-1} = 0, \mathcal{D}^m_{v,i}, \delta) = 1 \; - \; exp(-\lambda \delta(a^m_i, v) \sum_{u \in \mathcal{D}^m_{v,i}} \sum_{\substack{w \in N_{DN}(u) \; \cap \\ w \in act^{\mathcal{D}^m_v}(u)}} (T_{w,u} I_w)^T S_v) \tag{6}$$

where $act^{\mathcal{D}^m_v}$(u) denotes the list of activated users prior to user u's activation, with respect to m.

As shown in Figure 1a we can see that the first order Markov chain contains only the effect of all users with in the cascade context for v. As shown in Figure 1b, we added an extra layer of user influence modeling the trust factor as described in Equation (6).

Considering independent cascades, the likelihood of all cascades **C** is a product of the likelihoods given by Equation (2)

$$\mathcal{L}(C) = \prod_{m=1}^{|C|} \prod_{v \in V} P(\mathbf{z}^m_v | \delta) \tag{7}$$

We learn the parameter of the model, namely I and S latent feature vectors by minimizing the negative of logarithm of $\mathcal{L}(C)$ as follows:

$$\mathcal{L}(C) = - \sum_{m=1}^{|C|} \sum_{v \in V} \sum_{i=1}^{N} log \; p(z^m_{v,i} | z^m_{v,o-1}, \mathcal{D}^m_{v,i}, \delta) \tag{8}$$

4 Parameter Estimation

4.1 Problem Transformation

The parameter estimation problem aims to estimate the influence vector I_u and the susceptibility vector S_u for each user u. Directly minimizing the negative logarithmic likelihood function in Equation (8) corresponds to the maximum likelihood estimation (MLE) method, which encounters two problems in practical implementation. First, one cascade context (regarding a specific user) could appear in many different cascades, which would incur large computation overhead due to duplicate computations. Second, directly maximizing likelihood may result in model overfitting, which is a common issue for the MLE method.

We address the above issue in two steps. In the first step, we transform the per-cascade (message) likelihood function in Equation (8) into an aggregated per-cascade context likelihood function. For simplicity of illustration, we define the following notations. Define $\mathcal{P}_v = \{\mathcal{D}_{v,i}\}$ as the set of all possible cascade contexts regarding user v in the input dataset. Define $n_{z_{v,i}, \mathcal{D}_{v,i}}$ as the number of cascades among all cascades which contain the cascade context $\mathcal{D}_{v,i}$ and end up with user state $z_{v,i}$. The transformed negative logarithmic likelihood function is defined as follows:

$$\hat{\mathcal{L}}(\mathbf{C}) = - \sum_{v \in V} \sum_{\mathcal{D}_{v,i} \in \mathcal{P}v} \left(n_{z_{v,i}, \mathcal{D}_{v,i}} \log p(z_{v,i} | z_{v,i-1}, \mathcal{D}_{v,i}, \delta) \right) \tag{9}$$

In the second step, we add regularization into the loss function to avoid the overfitting problem of MLE. Specifically, we use two types of regularization. First, we use the influence and susceptibility matrices independently, and regularize the likelihood function using the Frobenius norm ($\| \cdot \|_F$) of the matrices respectively. We use γ_I and γ_S to denote the weight factor for the influence matrix and the susceptibility matrix respectively. Next, we further consider the similarity between influence and susceptibility of the same user. As mentioned in [1], when the number of latent dimension is properly selected, each latent feature can correspond to the influence and susceptibility of a certain topic. Intuitively, if a user v is interested in certain topics, he/she would mainly retweet messages about

those topics, which makes the corresponding entries in S_v larger than others. Then the probability for the followers of this user to retweet those topics is larger than other topics, which makes those entries of I_v relatively large. This implies that the direction of S_v and I_v should be similarity, therefore, we add the regularization term $(\|I_v\|_2 \|S_v\|_2 - I_v \cdot S_v)$. The final parameter estimation problem is as follows:

$$\begin{aligned} \max \quad & \hat{\mathcal{L}}(\mathbf{C}) = -\sum_{v \in V} \sum_{\mathcal{D}_{v,i} \in \mathcal{P}v} \left(n_{z_{v,i}, \mathcal{D}_{v,i}} \log p(z_{v,i} | z_{v,i-1}, \mathcal{D}_{v,i}, \delta) \right) \\ & \qquad + \gamma_I \|I\|_F^2 + \gamma_S \|S\|_F^2 + \phi \sum_v (\|I_v\|_2 \|S_v\|_2 - I_v \cdot S_v) \\ \text{s.t.} \quad & I_{ij} \geq 0, S_{ij} \geq 0, \forall i, j \end{aligned} \tag{10}$$

4.2 Optimization Design

It can be noted that the objective function in Eq. (10) is a convex function, hence the problem is minimization of a convex function subject to linear constraints. Many convex optimization algorithms can be used to solve the problem, for example, interior point methods like the barrier method. In this report, we use the Gradient Projection (GP) method [**?**], specifically due to its capability in decomposition-based parallel implementation for reducing computation time, to be elaborated in the following.

We first derive the gradient vectors of the objective function, as follows:

$$\begin{aligned} \frac{\partial \hat{\mathcal{L}}}{\partial I_u} = & -\lambda \sum_{v \in V} S_v \sum_{\mathcal{D}_{v,i} \in \mathcal{P}(v)} \mathcal{I}_{u \in \mathcal{D}_{v,i}} \left(n_{z_{v,i}=1, \mathcal{D}_{v,i}} \frac{1 - p_{v, \mathcal{D}_{v,i}}}{p_{v, \mathcal{D}_{v,i}}} - n_{z_{v,i}=0, \mathcal{D}_{v,i}} \right) \\ & + \gamma_I I_u + \phi \left(\frac{\|S_u\|_2}{\|I_u\|_2} I_u - S_u \right) \end{aligned} \tag{11}$$

$$\begin{aligned} \frac{\partial \hat{\mathcal{L}}}{\partial S_v} = & -\lambda \sum_{\mathcal{D}_{v,i} \in \mathcal{P}(v)} \sum_{u \in \mathcal{D}_{v,i}} I_u \left(n_{z_{v,i}=1, \mathcal{D}_{v,i}} \frac{1 - p_{v, \mathcal{D}_{v,i}}}{p_{v, \mathcal{D}_{v,i}}} - n_{z_{v,i}=0, \mathcal{D}_{v,i}} \right) \\ & + \gamma_S S_v + \phi \left(\frac{\|I_v\|_2}{\|S_v\|_2} S_v - I_v \right) \end{aligned} \tag{12}$$

Observe that both in the original objective function and in the two gradient functions as above, the first term is the sum of certain functions over all (u, v) pairs where there exists some cascade context $\mathcal{D}_{v,i}$ such that $u \in \mathcal{D}_{v,i}$. Therefore the computation of the first term can be naturally decomposed into computations per such (u, v) pair, which can be paralleled among computation entities as long as the corresponding parameter vectors I_u and S_v are shared among all entities. The rest terms in the functions cannot be decomposed.

The gradient vectors for the model with the Trust factor included changes to:

$$\begin{aligned} \frac{\partial \hat{\mathcal{L}}}{\partial I_u} = & -\lambda \sum_{v \in V} S_v \sum_{\substack{w \in N_{DN}(v) \cap \\ w \in act^{\mathcal{D}_v^m}(v)}} T_{w,v} \sum_{\mathcal{D}_{v,i} \in \mathcal{P}(v)} \mathcal{I}_{u \in \mathcal{D}_{v,i}} \left(n_{z_{v,i}=1, \mathcal{D}_{v,i}} \frac{1 - p_{v, \mathcal{D}_{v,i}}}{p_{v, \mathcal{D}_{v,i}}} - n_{z_{v,i}=0, \mathcal{D}_{v,i}} \right) \\ & + \gamma_I I_u + \phi \left(\frac{\|S_u\|_2}{\|I_u\|_2} I_u - S_u \right) \end{aligned} \tag{13}$$

$$\begin{aligned} \frac{\partial \hat{\mathcal{L}}}{\partial S_v} = & -\lambda \sum_{\mathcal{D}_{v,i} \in \mathcal{P}(v)} \sum_{u \in \mathcal{D}_{v,i}} \sum_{\substack{w \in N_{DN}(u) \cap \\ w \in act^{\mathcal{D}_v^m}(u)}} (T_{w,u} I_w) \left(n_{z_{v,i}=1, \mathcal{D}_{v,i}} \frac{1 - p_{v, \mathcal{D}_{v,i}}}{p_{v, \mathcal{D}_{v,i}}} - n_{z_{v,i}=0, \mathcal{D}_{v,i}} \right) \\ & + \gamma_S S_v + \phi \left(\frac{\|I_v\|_2}{\|S_v\|_2} S_v - I_v \right) \end{aligned} \tag{14}$$

Algorithm 1: Semi-distributed GP method for parameter estimation

Input: Cascade dataset, regularization weights γ_I, γ_S and ϕ, maximum iterations M, convengence threshold ϵ.

Output: Influence matrix I and susceptibility matrix S

[Central]: Initialize I and S as random matrices with dimension $n \times n$, where n is the number of nodes in the dataset;

[Central]: Initialize $m \times 1$ vector $lossEpoch$ to store the value of loss function for each iteration, set $convergence = False$;

do

  [Central]: Assign tasks to each agent;

  `// Start of parallel computation`

  for *each local agent* **do**

    [Local]: Calculate the first terms in Eq. (13) and (14);

  end

  [Central]: Simultaneously calculate the gradients of regularization terms in Eq. (13) and (14);

  `// End of parallel computation`

  [Central]: Aggregate gradients, update I and S using GP method;

  [Central]: Calculate the value of loss function by Eq. 10 with updated I and S and push the result into $lossEpoch$;

  [Central]: If $lossEpoch$ is decreased less than ϵ, set $convergence = True$.

while *not convergence and maximum iterations not reached*;

return *final I and S.*

Therefore, we propose the following semi-distributed algorithm for the above optimization problem, as shown in Algorithm 1.

There are two levels of agents to carry out the algorithm. The central agent is responsible for dispatching computation tasks, carrying out non-decomposable computations, and aggregates computation results. The local agents are responsible for conducting distributed computation based on central agent's assignment. The algorithm starts with randomly assigned parameter vectors (I_u and S_v). In each iteration, the algorithm calculates the gradients regarding the current parameters (in distributed manner). It then updates the corresponding parameters with the projected gradients (projection in order to enforce non-negativity constraints). The algorithm stops either at convergence or when reaching the maximum number of iterations allowed.

5 Performance Evaluation

5.1 Experimental Setup

To evaluate the performance of our model and compare it with the previous ones, we learn the influence matrix I and the susceptibility matrix S in training phase and then apply the corresponding rows in these two matrices to computation of influence probability for the testing samples. We split all the samples for each user into 10 folds and use 9 folds for training and 1 fold for testing. This provides the model information of retweeting behavior for all users who had ever been activated by his/her neighbors so that the model can be generalized to accomplish predictions on testing samples. In the training phase each element of I and S is initialized by sampling from normal distribution. By solving the optimization problem explained in Section 4, I and S.

We compare the prediction results of our model (LIS-similarity) to the LIS model introduced by [1] and random guess.

LIS: In this model, the influence and susceptibility matrices are learnt without taking similarity into account.

Random guess: This approach directly applies the randomly initialized influence and susceptibility matrices for prediction. The performance of this approach can show whether the probabilistic model introduced in Section 3 is reasonable on the datasets or not.

Given I and S, the influence probability of testing samples can be computed through the probabilistic model $p(z_{v,i}|z_{v,i-1}, \mathcal{D}_{v,i}, \delta)$ (see algorithm 2). With the predicted influence probability of each testing sample, a moving probability threshold is applied to get the data points for the ROC (Receive Operating Characteristics) curve. Each data point in ROC curve is a tuple: (TPR, FPR). The true positive rate is $\frac{TP}{P}$ and the false positive rate is $\frac{FP}{N}$ where TP is the number of true positives, FP is the number of false positives, P is the number of positive samples and N is the number of negative samples in the testing set. As the threshold moves from 1 to 0, both FPR and TPR increases for the positive class and decreases for the negative class. Finally the result is reported by the AUC (Area under Curve) value. The AUC value of ROC curve is between 0 and 1, the larger the better.

Algorithm 2: Semi-distributed computation of influence probability

Input: Testing cascade dataset, Influence matrix I and susceptibility matrix S, regularization weights γ_I, γ_S and ϕ

Output: Retweeting probability map $probMap << Int > NodeIndex, < ArrayList < Double >> Probabilities >$

1 **[Central]**: Read I and S from results of algorithm 1;
2 **[Central]**: Initialize $probMap$ with empty arraylist entries, one for each node;
3 **[Central]**: Assign tasks to each agent;
`// Start of parallel computation`
4 **for** *each local agent* **do**
5 **[Local]**: Calculate the retweeting probability by Eq.(4) for each combination of given (node under influence, social context);
6 **[Local]**: Push the result into corresponding entry in $probMap$;
7 **end**
`// End of parallel computation`
8 **return** *probMap*

5.2 Dataset Description

We prepare two datasets for evaluation of the models. We collect the Weibo dataset from authors of [1] and the Twitter dataset from authors of [10]. For the Twitter dataset, we sampled it for our task by removing those users with less than 10 retweets such that all the users can appear at least once in both training and testing set.

As shown in Fig. 2, each entry of either of the datasets composed of four parts separated by the comma and hash symbol. The first part indicates the unique index of the user under influence. The second part is the binary variable $z_{v,i} \in \{1, 0\}$, indicating the status of this user after being influenced by the active neighbors. $z_{v,i} = 0$ means that this user had never retweeted under influence of the social context for any message, keeping the unactivated status. and $z_{v,i} = 1$ means that the user under influence from the social context retweeted. The third part is the social context, it shows a list of previous active neighbors in chronological order. Finally, the fourth path shows how many times the combination of the three previous parts repeated. In the example shown in Fig. 2, the situation that user 227 did not retweet under influence of users $\{0, 6, 8, 2, 20\}$ repeated 12 times in the dataset.

To show the Weibo dataset and Twitter dataset are good samples such that the experimental results can be generalized to other datasets, we compute the degree distribution for these two datasets here. We show the histogram of degree distribution for our Weibo and Twitter dataset separately (see Fig. 3).

Besides the degree distribution, we also carry out an empirical study where we show the correlation between influence probability and number of active numbers (see Fig. 4). The result justifies the non-negative constraints on I and S in our problem formulation. As the retweeting probability increases with the number of active neighbors, which means the inner product should be non-negative of any I_u and S_v for $u \neq v$.

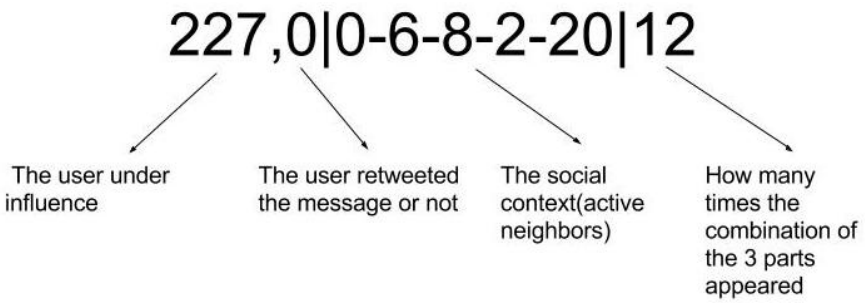
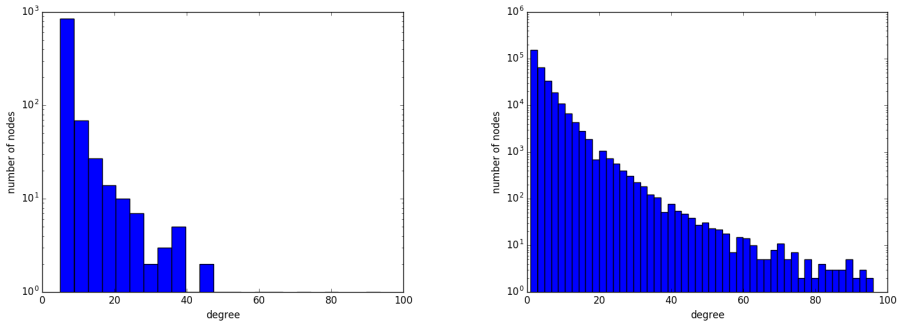


Figure 2: An example entry of the data used in training and testing the models.



(a) Degree distribution for the Weibo dataset (b) Degree distribution for the Twitter dataset

Figure 3: Social network statistics for the two datasets used in our experiments.

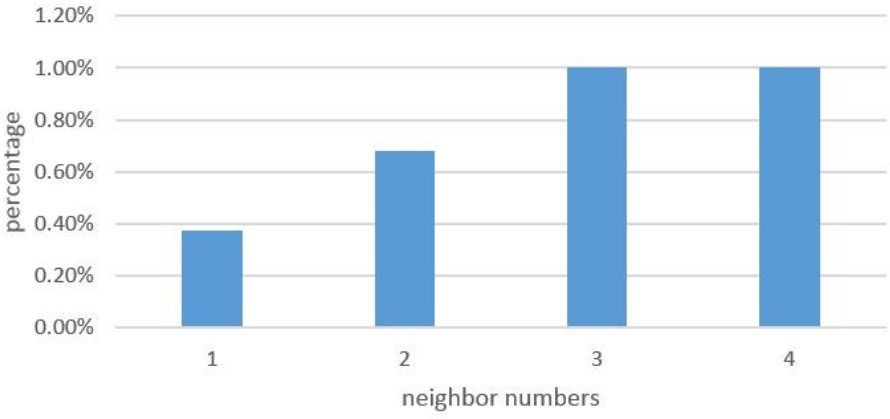


Figure 4: Correlation between number of active neighbors and the retweeting probability. This observation justifies the non-negative constraints on I and S.

Model	Weibo	Twitter
LIS-similarity	0.371	0.988
LIS	0.249	0.787

Table 1: Comparison of average cosine similarity of influence and susceptibility matrix learnt by LIS-simialrity and LIS.

5.3 Experimental Results

The prediction results are shown in Fig 5 as ROC curves and their corresponding AUC values. For the Weibo dataset, the LIS-similarity model performs 1% better than LIS model, while both of them significantly outperform the random guess by at least 23%. For the Twitter dataset, the proposed model outperforms others by at least 3%, however, the improvement of LIS model over random guess is trivial (1%). To examine if the added regularization term ($\|I_v\|_2 \|S_v\|_2 - I_v \cdot S_v$) really pushed the direction of I_v and S_v to be similar for every user v in the output of algorithm 1, we also monitor the average cosine similarity calculated as below:

$$cossim(I, S) = \frac{1}{n} \sum_{v \in V} \frac{I_v S_v}{\|I_v\|_2 \|S_v\|_2} \tag{15}$$

In table 1, it is shown that average cosine similarity computed by I and S estimated by LIS-similarity is significantly greater than that of those learnt by LIS for both of the datasets. In addition, out of our surprise, the modified objective function in LIS-simialrity results in non-trvial boost in convergence rate (in terms of run time) for the model compared to the original LIS model. We will keep the analysis of this observation for future work.

6 Conclusions and Future Work

In this project we formulate the problem of modeling retweeting probability as a probabilistic model which takes inner product of user-specific influence and susceptibility vectors as input. The model is trained by a maximum likelihood estimation process where a constrained convex optimization problem is solved by Gradient Projection method. From the perspective of implementation, we adopt the multi-threading framework from authors of [1] and extend it to perform the training and testing phase for the LIS-similarity model. The experimental results show that the proposed LIS-similarity model outperforms LIS and random guess in both datasets in terms of AUC value. For future work, another aspect of understanding social influence, namely temporal dynamics, is not covered by this model but plays an important role in determination of retweeting probability. A potentially more predictive model should be able to distinguish cases happening in early stage of cascades from those in final stage. As far as the probability model is concerned, there can be more improvements over our model. Instead of considering a first order Markov chain for modeling the status vector of a user, one can use advanced hierarchical bayesian modeling to formulate the current state of the user based on its previous interactions with the active users. Similarly with respect to the trust propagation technique, one can use ensemble models for the trust factor as has been shown in [12] which outperforms the existing models in recommendation systems.

Paperlist

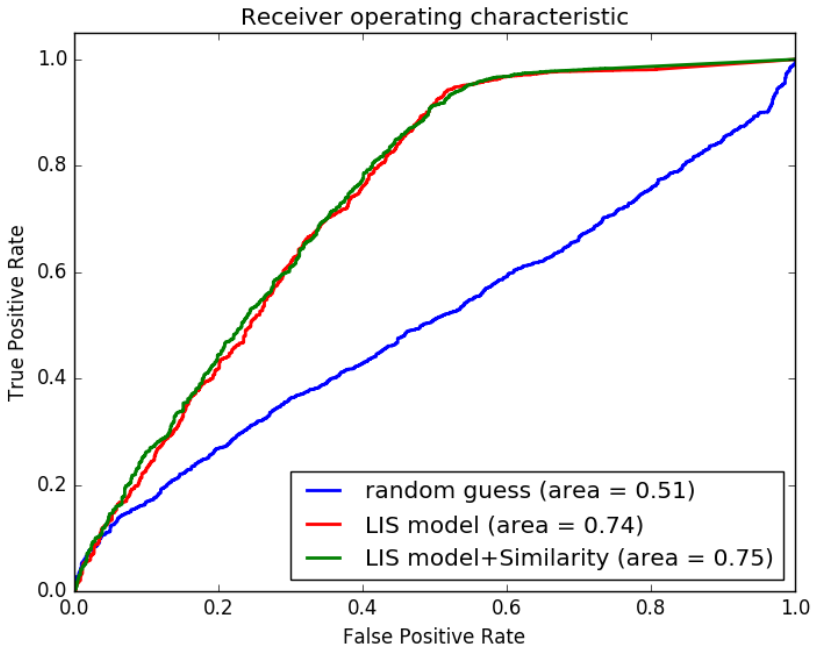
[1] Wang, Yongqing, et al. "Learning user-specific latent influence and susceptibility from information cascades." Proc. of AAAI, 2015.

[2] Liu, Shenghua, et al. "Learning Sentimental Influences from Users' Behaviors." arXiv preprint arXiv:1608.03371 (2016).

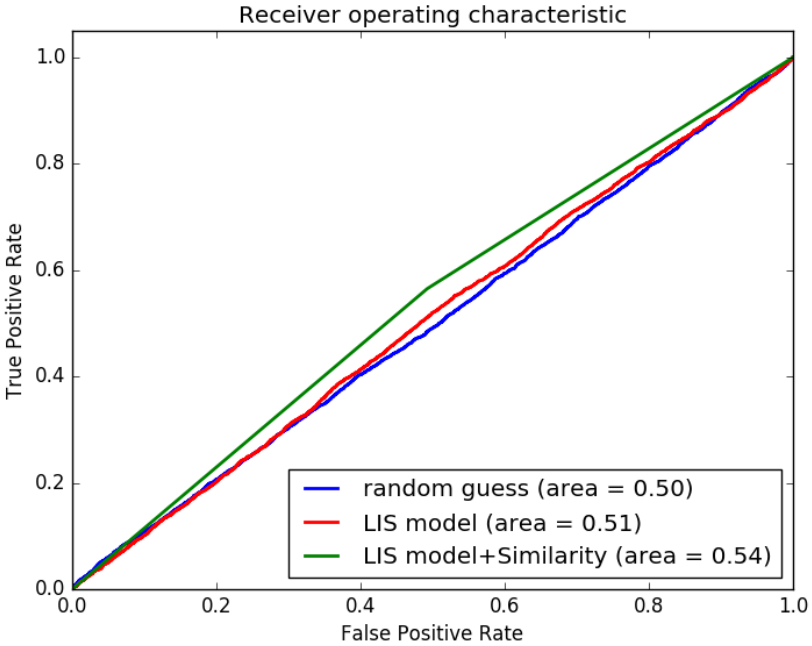
[3] Zhang, Jing, et al. "Social Influence Locality for Modeling Retweeting Behaviors." IJCAI. Vol. 13. 2013.

[4] Goyal, Amit, Francesco Bonchi, and Laks VS Lakshmanan. "Learning influence probabilities in social networks." Proc. of ACM WSDM, 2010.

[5] Lee, Kyumin, et al. "Who will retweet this? detecting strangers from twitter to retweet information." ACM TIST 6.3 (2015): 31.



(a) ROC curve for the Weibo dataset



(b) ROC curve for the Twitter dataset

Figure 5: Retweeting behavior prediction results

[6] Myers, Seth A., Chenguang Zhu, and Jure Leskovec. "Information diffusion and external influence in networks." Proc. of ACM SIGKDD, 2012.

[7] Ugander, Johan, et al. "Structural diversity in social contagion." Proceedings of the National Academy of Sciences 109.16 (2012): 5962-5966.

[8] Amit Goyal, Francesco Bonchi, Laks V. S. Lakshmanan, Learning Influence Probabilities in Social Networks. In Proc. of the 3rd ACM International Conference on Web Search and Data Mining, WSDM 2010, New York City, 2010.

[9] Saito, Kazumi, Ryohei Nakano, and Masahiro Kimura. "Prediction of information diffusion probabilities for independent cascade model." International Conference on Knowledge-Based and Intelligent Information and Engineering Systems. Springer Berlin Heidelberg, 2008.

[10] Lilian Weng, Filippo Menczer, and Yong-Yeol Ahn. Predicting Meme Virality in Social Networks using Network and Community Structure. To appear in 8th AAAI Intl. Conf. on Weblogs and social media (ICWSM). 2014.

[11] Mohsen Jamali, Martin Ester. A Matrix Factorization Technique with Trust Propagation for Recommendation in Social Networks. Proceedings of RecSys 2010, Barcelona, Spain.

[12] Hao Ma, Irwin King, Michael Lyu. Learning to recommend with Social Trust Ensemble. In Proceedings of SIGIR 2009, Boston, Massachusetts, USA.