Social Action Tracking via Noise Tolerant Time-varying Factor Graphs

Chenhao Tan[†], Jie Tang[†], Jimeng Sun[‡], Quan Lin[♮*] and Fengjiao Wang[§*]
[†]Department of Computer Science and Technology, Tsinghua University, China
[‡]IBM TJ Watson Research Center, USA [♮]Huazhong University of Science and Technology, China
[§]Beijing University of Aeronautics and Astronautics, China

ABSTRACT

Users' behaviors (actions) in a social network are influenced by various factors such as personal interests, social influence, and global trends. However, few publications systematically study how social actions evolve in a dynamic social network and to what extent different factors affect the user actions.

In this paper, we propose a Noise Tolerant Time-varying Factor Graph Model (NTT-FGM) for modeling and predicting social actions. NTT-FGM simultaneously models social network structure, user attributes and user action history for better prediction of the users' future actions. More specifically, a user's action at time t is generated by her latent state at t, which is influenced by her attributes, her own latent state at time $t-1$ and her neighbors' states at time t and $t-1$. Based on this intuition, we formalize the social action tracking problem using the NTT-FGM model; then present an efficient algorithm to learn the model, by combining the ideas from both continuous linear system and Markov random field.

Finally, we present a case study of our model on predicting future social actions. We validate the model on three different types of real-world data sets. Qualitatively, our model can discover interesting patterns of the social dynamics. Quantitatively, experimental results show that the proposed method outperforms several baseline methods for social action prediction.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Data Mining; J.4 [**Computer Applications**]: Social and Behavioral Sciences

General Terms

Algorithms, Experimentation

Keywords

Social action tracking, Time-varying factor graphs, Social influence analysis

*This work was done when the last two authors are visiting Tsinghua University.

KDD'10, July 25–28, 2010, Washington, DC, USA.

1. INTRODUCTION

With the success of many large-scale online social networks, such as Facebook, MySpace, Ning, and Twitter, social network analysis has become a popular research topic, attracting tremendous interests from mathematics, biology, physics, computer science, and sociology. Considerable research has focused on finding the macro-level mechanisms of the social influence such as degree distributions, diameter, clustering coefficient, communities, and small world effect [1, 8, 21, 28]. However, these methods provide us with limited insights into the micro-level dynamics of the social network such as how an individual user changes his behaviors (actions) and how a user's action influences his friends.

It is well recognized that users' actions in a social network are influenced by various complex and subtle factors [11, 17]. In this work, we address the social action tracking problem: i.e., how to simultaneously model the social network structure, user attributes and user actions over time?

Recently, quite a few related studies have been conducted, for example, dynamic social network analysis [12, 16, 19, 24, 25], social influence analysis [2, 6, 7, 15, 23, 29, 20], and group behavior analysis [3, 13, 26, 31]. The social action tracking problem addressed in this paper is very different from these works. Dynamic social network analysis is to model how friendships drift over time using a dynamic model [24] or to investigate how different preprocessing decisions and different network forces such as selection and influence affect the modeling of dynamic networks [25]. Social influence analysis either aims to verify the existence of social influence [2, 6, 7, 15, 23] or tries to quantify the strength of the influence [10, 29]. Group behavior analysis intends to study the patterns of user joining different communities [26], or to learn the classification patterns based on the network structure and content information [31], or to study the statistics of user groups [3]. In the social action tracking problem, we try to model the various factors that may influence users' dynamic behaviors (actions) into a unified model.

Motivating Examples To clearly motivate this work, we conduct the following analysis on three real social networks: Twitter[1], Flickr[2], and Arnetminer[3]. On Twitter, we define the action as whether a user discusses the topic "Haiti Earthquake" on his microblogs (tweets). On Flickr, we define the action as whether a user adds a photo to his favorite list. On Arnetminer, the action is defined as whether a researcher publishes a paper on a specific conference (or journal). The analysis includes three aspects: (1) social influence; (2) time-dependency of users' actions; (3) action

[1]http://www.twitter.com, a microblogging system.
[2]http://www.flickr.com, a photo sharing system.
[3]http://arnetminer.org, an academic search system.

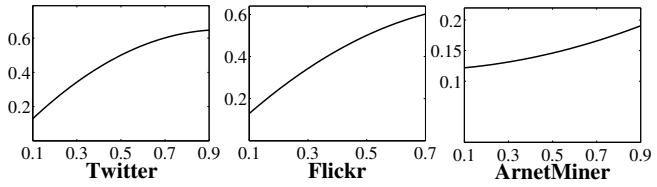


Figure 1: Social influence. The x-axis stands for the percentage of one's friends who perform an action at $t-1$ and the y-axis represents the likelihood that the user also performs the action at t.

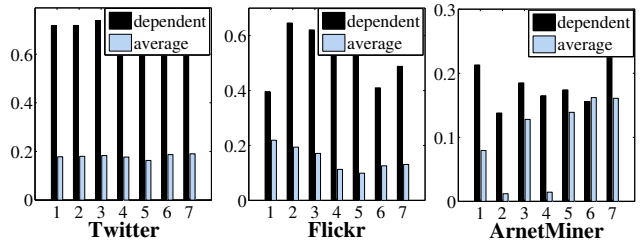


Figure 2: Time-dependency of users' actions. The x-axis stands for different timestamps. "dependent" denotes the likelihood that a user performs an action which was previously performed by herself; "average" represents the likelihood that a user performs the action.

correlation between users. Figure 1 shows the effect of social influence. We see that with the percentage of one's friends performing an action increasing, the likelihood that the user also performs the action is increased. For example, when the percentage of one's friends discussing "Haiti Earthquake" on their tweets increases the likelihood that the user posts tweets about "Haiti Earthquake" is also increased significantly. Figure 2 illustrates how a user's action is dependent on his historic behaviors. It can be seen that a strong time-dependency exists for users' actions. For instance, on Twitter, averagely users who posted tweets about "Haiti Earthquake" will have a much higher probability (+20-40%) to post tweets on this topic than those who never discussed this topic on their blogs. Figure 3 shows the correlation between users' actions at the same timestamp. An interesting phenomenon is that friends may perform an action at the same time. E.g., on Twitter, two friends have a higher probability (+19.6%) to discuss "Haiti Earthquake" than two users randomly chosen from the network.

Thus, the problem becomes how to effectively and efficiently track the dynamic users' actions. This problem is non-trivial and poses a set of unique challenges.

First, the social network data (e.g., network structure and social actions) are very noisy. Users performing the same action may not have the same preference towards that action. Likewise, users who did not perform the action do not mean they have no interests towards the action. Second, user behaviors are highly time-dependent. For example, the influence of a user on another (strongly) depends on their historic interactions. Third, users' actions are usually correlated. In addition, as real social networks are getting larger with thousands or millions of users. It is important to develop the model that can scale well to real large data sets.

Contributions In this paper, we try to systematically investigate the problem of social action tracking with the following contributions:

- We formally formulate the problem of social action tracking and propose a unified model: Noise Tolerant Time-varying Factor Graph Model (NTT-FGM).
- We present an efficient algorithm for model learning and develop a distributed implementation based on MPI (Message-Passing Interface) to scale up to real large networks.
- We present a case study on social action prediction using the learned NTT-FGM model.
- We conduct experiments on three different data sets: Twitter, Flickr, and Arnetminer. Experimental results show that the proposed NTT-FGM model can achieve a better performance for the action prediction than several alternative models.

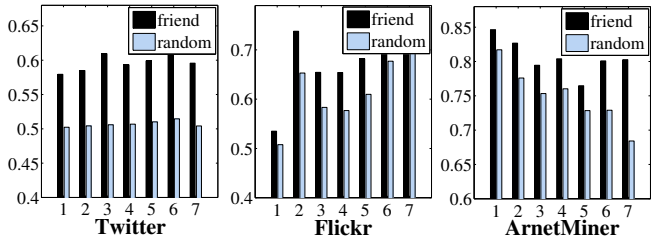


Figure 3: Action correlation. The x-axis stands for different time windows. "friend" denotes the likelihood that two friends perform an action together; "random" represents the likelihood that two random users perform the action together.

The rest of the paper is organized as follows: Section 2 formally formulates the problem; Section 3 explains the proposed model. Section 4 presents the algorithm for learning the model. Section 5 gives experimental results that validate the effectiveness and the computational efficiency of our methodology. Finally, Section 6 discusses related work and Section 7 concludes.

2. PROBLEM DEFINITION

In this section, we first give several necessary definitions and then present a formal definition of the problem.

A static social network can be represented as $G = (V, E)$, where V is the set of $|V| = N$ users and $E \subset V \times V$ is the set of directed/undirected links between users. Given this, we can define the user's action as follows.

Definition 1. **Action**: An action y performed by user v_i at time t can be represented as a triple (y, v_i, t) (or shortly y_i^t). Let Y^t be the set of actions of all users at time t. Further we denote all users' actions as the action history $\mathbf{Y} = \{(y, v_i, t)\}_{i,t}$.

Without loss of generality, we first consider the binary action, that is $y_i^t \in \{0, 1\}$, where $y_i^t = 1$ indicates that user v_i performed an action at time t, and $y_i^t = 0$ indicates that the user did not perform the action. Such an action log can be available from many online systems. For example, on Twitter, the action y_i^t can be defined as whether user v_i posts a tweet (microblog) about a specific topic (e.g., "Haiti Earthquake") at time t. Further, we assume that each user is associated with a number of attributes and thus have the following definition.

Definition 2. **Time-varying attribute matrix**: Let X^t be an $N \times d$ attribute matrix at time t in which every row $\mathbf{x}_i$ corresponds to a user, each column an attribute, and an element x_{ij} is the j^{th} attribute value of user v_i.

The attribute matrix describes user-specific characteristics, and can be defined in different ways. For example, on Twitter, each attribute can be defined as a keyword and the value of an attribute can be defined as the frequency of a keyword occurring on a user's

posted tweets. Thus, we can define the input of our problem, a set of attribute augmented networks.

Definition 3. **Attribute augmented network**: The attribute augmented network is denoted as $G^t = (V^t, E^t, X^t, Y^t)$, where V^t is the set of users and E^t is the set of links between users at time t, and X^t represents the attribute matrix of all users in the network at time t, and Y^t represents the set of actions of all users at time t.

Based on the above concepts, we can define the problem of social action tracking. Given a series of T time-dependent attribute augmented networks, the goal is to learn a model that can best fit the relationships between the various factors and the user actions. More precisely,

Problem 1. **Social action tracking.** Given a series of T time-dependent attribute augmented networks $\{G^t = (V^t, E^t, X^t, Y^t)\}$, where $t \in \{1, \cdots, T\}$, the goal of social action tracking is to learn a mapping function

$$f : (\{G^1, \ldots, G^{T-1}\}, V^T, E^T, X^T) \rightarrow Y^T$$

Note that in this general formulation, we allow the graph structure to evolve over time and also arbitrary dependency from the past. To have a tractable problem to work with, we model the time-dependency by introducing a latent state for each user. More specifically, their actions are generated by their latent states, which are dependent on their neighbors' states at time t and $t-1$.

Our formulation of social action tracking and prediction is quite different from existing work on dynamic social network analysis. Scripps et al. [25] investigate how different pre-processing decisions and different network forces such as selection and influence affect the modeling of dynamic networks. Sarkar and Moore [24] propose a dynamic model that accounts for friendships drifting over time. Both papers consider using user similarity and social structure to predict links between social users. Tang et al. study the topic-level social influence and Goyal et al. [10] investigate how to learn the influence probabilities from the history of users' actions. Both learned model can be used for action prediction. However, the methods do not consider user's own attributes and historic actions. One's action should be determined by his intrinsic preference to some extent. As for social influence analysis, there are quite a lot of publications focusing on measuring the existence of influence qualitatively [2, 27], but most of these methods do not consider modeling and predicting user actions.

3. NOISE TOLERANT TIME-VARYING FACTOR GRAPH MODEL (NTT-FGM)

To summarize, for modeling and tracking social actions, we have the following intuitions:

1. Users' actions at time t are influenced by their friends' historic actions (time $< t$).
2. Users' actions at time t are usually dependent on their previous actions.
3. Users' actions at a same time t have a (strong) correlation.

Moreover, the discrete variable y_i^t only models the user's action at a coarse level, but cannot describes the intention degree of the user to perform an action. Directly modeling the social actions Y would inevitably introduce noise to the model. Hence, a continuous variable for modeling the *action bias* is favorable.

With the intuitions discussed above, we propose a noise tolerant time-varying factor graph model (NTT-FGM) for social action tracking. Before explaining the model in detail, we first introduce the definition of latent action state.

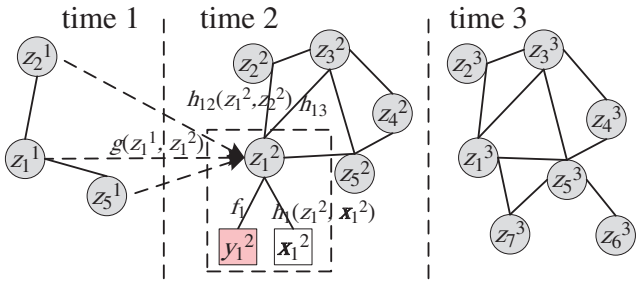


Figure 4: Graphical representation of the NTT-FGM model. Each circle stands for a user's latent action state z_i^t at time t in the network, which is used to characterize the intention degree of the user to perform the action; the latent state is associated with the action y_i^t, a vector of attributes $\mathbf{x}_i^t$, and depends on friends' historic actions $\mathbf{z}_{\sim v_i}^{t-1}$ and correlates with friends' actions $\mathbf{z}_{\sim v_i}^t$ at time t; $g(.)$ denotes a factor function to represent the friends' influence on a user's action; $h_i(.)$ represents a factor defined on user v_i's attributes; and $h_{ij}(.)$ represents a factor to capture the correlation between users' actions.

Definition 4. **Latent action state**: For each user's action y_i^t, we define a (continuous) latent state $z_i^t \in [0, 1]$, which corresponds to a combination of the observed action y_i and a possible bias, to describe the actual intention degree of the user to perform the action.

Figure 4 shows the graphical structure of the NTT-FGM model. An action of user v_i at time t, i.e., y_i^t is modeled by using a (continuous) latent action state z_i^t, which is dependent on friends' historic actions $\mathbf{z}_{\sim v_i}^{t-1}$ (where $\sim v_i$ represents friends of user v_i in the network), users' action correlation $\mathbf{z}_{\sim v_i}^t$, and users' attributes $\mathbf{x}_i^t$. Specifically, in the NTT-FGM model, each discrete action is mapped into the latent state space and the action bias is modeled using a factor function. For example, for $y_i^t = 1$, a small value of its corresponding z_i^t suggests that a user v_i has a low intention to perform the action, thus a large action bias $|y_i^t - z_i^t|$. Next, influence between users is modeled using the latent states based on the same assumption as in HMM [9] and Kalman Filters [14]: latent states of users' actions at time t are conditionally independent of all the previous states given the latent states at time $t-1$. Finally, actions' correlation is also modeled in the latent state space. A Markov random field is defined to model the dependency (correlation) among the continuous latent states. Different from the traditional Markov random field model (e.g., CRF [18], HMM [9], Kalman Filters [14]), the NTT-MRF model uses a continuous variable to describe the latent state, and utilizes a combination of multivariate Gaussian function and Markov random field to incorporate both time-inter and time-intra dependency between users' actions.

Now, we explain the proposed NTT-FGM model in detail. Given a series of attribute augmented networks $\mathbf{G} = \{G^t = (V^t, E^t, X^t, Y^t)\}$, $t \in \{1, \cdots, T\}$ and $V = V^1 \cup V^2 \cup \ldots \cup V^T, |V| = N$, we can define the joint distribution over the actions $\mathbf{Y}$ given $\mathbf{G}$ as

$$p(\mathbf{Y}|\mathbf{G}) = \prod_{t=1}^{T} \prod_{i=1}^{N} f(y_i^t|z_i^t) f(z_i^t|\mathbf{z}_{\sim v_i}^{t-1}) f(z_i^t|\mathbf{z}_{\sim v_i}^t, \mathbf{x}_i^t) \qquad (1)$$

where notation $\sim v_i$ represents neighbors of v_i in the social network. The joint probability has three types of factor functions, corresponding to the intuitions we have discussed. Specifically,

- Action bias factor: $f(y_i^t|z_i^t)$ represents the posterior probability of user v_i's action y_i at time t given the continuous latent state z_i^t;
- Influence factor: $f(z_i^t|\mathbf{z}_{\sim v_i}^{t-1})$ reflects friends' influence on user v_i's action at time t;
- Correlation factor: $f(z_i^t|\mathbf{z}_{\sim v_i}^t, \mathbf{x}_i^t)$ denotes the correlation between users' action at time t.

The three factors can be instantiated in different ways, reflecting our prior knowledge for different applications. In this paper, we will give a general definition for the three factors. For the action bias factor $f(y_i^t|z_i^t)$, we define it using a Gaussian function:

$$f(y_i^t|z_i^t) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{(y_i^t - z_i^t)^2}{2\sigma^2}\} \tag{2}$$

where σ is a variance to control the bias and its value can be learned using an EM-style algorithm or predefined empirically. Note that if we only consider the binary action, the bias factor can be also defined based on a Bernoulli distribution.

For influence factor $f(z_i^t|\mathbf{z}_{\sim v_i}^{t-1})$, we first define an binary $N \times N$ matrix M^{t-1} to describe the social network at time $t-1$, where the element $m_{ij}^{t-1} = 1$ represents that user v_i and v_j have a relationship in the social network (i.e., $e_{ij} \in E$), and $m_{ij}^{t-1} = 0$ indicates there is no relationship between v_i and v_j. Given this, we can formally define the influence factor as:

$$f(z_i^t|\mathbf{z}_{\sim v_i}^{t-1}) = \frac{1}{Z_1}\exp\{\sum_{j=1}^{N} \lambda_{ji} m_{ji}^{t-1} g_{ji}(z_i^t, z_j^{t-1})\} \tag{3}$$

where $g_{ji}(z_i^t, z_j^{t-1})$ is a function defined on the latent states of two users z_i^t and z_j^{t-1}; λ_{ji} (when $m_{ji}^{t-1} = 1$) represents the influence degree of v_j on v_i. For example, given a higher influence λ_{ji}, the action of user v_j is more likely to induce user v_i to behave in a similar way. Z_1 is a normalization factor. When $j = i$, we refer to the influence as self-influence, which actually characterizes the dependency of the user's action on his own previous state.

The correlation factor can be naturally modeled in a Markov random field. Therefore, by the fundamental theorem of random fields, we can define the correlation factor as:

$$\begin{aligned} f(z_i^t|\mathbf{z}_{\sim v_i}^t, \mathbf{x}_i) &= \frac{1}{Z_2} \exp\{(\sum_{j=1}^{N} \beta_{ij} m_{ij}^t h_{ij}(z_i^t, z_j^t) \\ &+ \sum_{k=1}^{d} \alpha_k h_k(z_i^t, x_{ik}^t))\} \end{aligned} \tag{4}$$

where $h_{ij}(z_i^t, z_j^t)$ is a feature function to capture the correlation between user v_i and v_j at time t; $h_k(z_i^t, x_{ik}^t)$ is a feature function defined on user v_i and the k-th attribute x_{ik}; d is the number of attributes; β_{ij} and α_k are respectively weights of the two functions; and Z_2 is again a normalization factor.

Finally, by integrating Eqs. (2)-(4) into Eq. (1), we can obtain the following joint probability

$$\begin{aligned} p(\mathbf{Y}|\mathbf{G}) = \frac{1}{Z}\exp\{&\sum_{t=1}^{T}\sum_{i=1}^{N} \frac{(y_i^t - z_i^t)^2}{2\sigma^2} + \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_{ij} m_{ji}^{t-1} g(z_i^t, z_j^{t-1}) \\ &+ \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} \beta_{ij} m_{ij}^t h_{ij}(z_i^t, z_j^t) + \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{d} \alpha_k h_k(z_i^t, x_{ik}^t)\} \end{aligned} \tag{5}$$

where $Z = (2\pi\sigma^2)^{\frac{N\times T}{2}} Z_1 Z_2$.

Learning NTT-FGM is to estimate a parameter configuration $\theta = (\{z_i\}, \{\alpha_k\}, \{\beta_{ij}\}, \{\lambda_{ij}\})$ from a given historic action log $\mathbf{Y}$, that maximizes the log-likelihood objective function $\mathcal{O}(\theta) = \log p_\theta(\mathbf{Y}|\mathbf{G})$, i.e.,

$$\theta^\star = \arg\max \mathcal{O}(\theta) \tag{6}$$

4. MODEL LEARNING

There are two challenges to solve the objective function. First, as the network structure in the social network can be arbitrary (may contain cycles), traditional methods such as Junction Tree [32] and Belief Propagation [34] cannot result in an exact solution. Second, to calculate the normalization factor Z, it is necessary to guarantee that the denominator of Eq. (5), i.e., the exponential function $\exp\{.\}$, is integrable. Based on these considerations, we instantiate the factor functions $g(.)$ and $h(.)$ as follows

$$g_{ji}(z_i^t, z_j^{t-1}) = -(z_i^t - z_j^{t-1})^2 \tag{7}$$

$$h_{ij}(z_i^t, z_j^t) = -(z_i^t - z_j^t)^2 \tag{8}$$

$$h_k(z_i^t, x_{ik}^t) = -(z_i^t - x_{ik}^t)^2 \tag{9}$$

We see that all of the factor functions are defined by quadratic functions. This is because quadratic equation satisfies the above two requirements: it is integrable and it offers the possibility to design an exact solution. Moreover, by defining in this way, the influence factor and the correlation factor can be elegantly explained with the information diffusion theory, by which the actions of users spread in the social network along the relationships [4, 12].

Finally, the objective function $\mathcal{O}(\theta)$ can be rewritten as

$$\begin{aligned} \mathcal{O}(\theta) = -\log Z - \{&\sum_{t=1}^{T}\sum_{i=1}^{N} \frac{(y_i^t - z_i^t)^2}{2\sigma^2} + \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_{ji} m_{ji}^{t-1} (z_i^t - z_j^{t-1})^2 \\ &+ \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} \beta_{ij} m_{ij}^t (z_i^t - z_j^t)^2 + \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{d} \alpha_k (z_i^t - x_{ik}^t)^2\} \end{aligned} \tag{10}$$

where

$$\begin{aligned} Z = C \int_y \int_z \exp\{&-\sum_{t=1}^{T}\sum_{i=1}^{N} \frac{(y_i^t - z_i^t)^2}{2\sigma^2} - \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_{ji} m_{ji}^{t-1} (z_i^t - z_j^{t-1})^2 \\ &- \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} \beta_{ij} m_{ij}^t (z_i^t - z_j^t)^2 - \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{d} \alpha_k (z_i^t - x_{ik}^t)^2\} dz dy \end{aligned} \tag{11}$$

where $C = (2\pi\sigma^2)^{\frac{N\times T}{2}}$ is a constant.

4.1 The Learning Algorithm

The task of model learning is to estimate the parameters $\theta = (\{z_i\}, \{\alpha_k\}, \{\beta_{ij}\}, \{\lambda_{ij}\})$ by solving the objective function Eq. (10). For this purpose, we need to first solve the integration of Z. As y is discrete, we can easily integrate out the first term in the $\exp\{.\}$ function of Eq. (11). Further to guarantee that Z is integrable, we must have $\alpha_k > 0, \beta_{ij} > 0, \lambda_{ij} > 0$. It is still difficult to solve the integration. To deal with this, our basic idea is to transform the exponential function $\exp\{.\}$ into a multivariate Gaussian distribution, and calculate the integration as follows: (Derivation is given in Appendix 10.1.)

$$Z = Const \cdot |A|^{-\frac{1}{2}} \exp\{\mathbf{b}^T A^{-1} \mathbf{b} - c\} \tag{12}$$

Input: number of iterations I and learning rate η;
Output: learned parameters $\theta = (\{z_i\}, \{\alpha_k\}, \{\beta_{ij}\}, \{\lambda_{ij}\})$;

Initialize $\mathbf{z} = \mathbf{y}$;
Initialize α, β, λ;
repeat
  E Step: % fix $\mathbf{z}$, learn α, β, λ;
  for $i = 1$ *to* I **do**
    Compute gradient $\nabla_{\log \alpha_k}, \nabla_{\log \beta_{ij}}, \nabla_{\log \lambda_{ij}}$;
    Update $\log \alpha_k = \log \alpha_k + \eta \times \nabla_{\log \alpha_k}$;
    Update $\log \beta_{ij} = \log \beta_{ij} + \eta \times \nabla_{\log \beta_{ij}}$;
    Update $\log \lambda_{ij} = \log \lambda_{ij} + \eta \times \nabla_{\log \lambda_{ij}}$;
  end
  M Step: % fix α, β, λ learn $\mathbf{z}$;
  Solve the following linear equation:

$$(A + \mathbf{I})\mathbf{z} = \mathbf{y} + X\alpha$$

until *convergence*;

Algorithm 1: Expectation maximization.

where $c = \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{k=1}^{d} \alpha_k x_{ik}^t$; $Const$ is a constant; A is a $NT \times NT$ block tridiagonal matrix; and $\mathbf{b} = \mathbf{X}\alpha$ is a NT-dimension vector and $\mathbf{X} = \{X^1 : X^2 : \cdots : X^T\}$is $NT \times d$ matrix by concatenating all time-varying attribute matrices together.

Given this, we can design an EM-style algorithm to maximize $\mathcal{O}(\theta)$, as summarized in Algorithm 1: (Details including gradients of the parameters are given in Appendix 10.2.)

- **E-step:** fix $\mathbf{z}$ and update all α, β, and λ, using a gradient descent method;
- **M-step:** fix α, β, and λ to update all $\mathbf{z}$, by solving a linear system.

4.2 Social Action Prediction

Based on the learned parameters θ, we can predict the users' future actions. Specifically, for predicting a user's action y_i^{T+1} at time $T+1$, we first compute the latent state z_i^{T+1}; and then use the latent state to infer the action y_i^{T+1}. To compute the latent state z_i^{T+1}, we have the following formula:

$$z_i^{T+1} = \frac{\sum_{k=1}^{d} \alpha_k x_{ik} + \sum_{j=1}^{N} \lambda_{ji} m_{ji}^T z_j^T}{\sum_{k=1}^{d} \alpha_k + \sum_{j=1}^{N} \lambda_{ji} m_{ji}^T} \tag{13}$$

However, the above equation calculates the latent state independently and ignore the correlation between actions. By further considering the action correlation factor, that is to compute all $\mathbf{z}$ together, we can solve the following linear system:

$$\begin{aligned} \forall\, i,\ & \sum_{k=1}^{d} \alpha_k (z_i^{T+1} - x_{ik}) + \sum_{j=1}^{N} \lambda_{ji} m_{ji}^T (z_i^{T+1} - z_j^T) \\ & + \sum_{j=1}^{N} \beta_{ij} (z_i^{T+1} - z_j^{T+1}) + \sum_{j=1}^{N} \beta_{ji} (z_i^{T+1} - z_j^{T+1}) = 0 \end{aligned} \tag{14}$$

Then, we can predict the users' actions y according to their corresponding latent states z by:

$$y_i^{T+1} = \begin{cases} 0 & \text{if } |z_i^{T+1} - \overline{z}_+| <= |z_i^{T+1} - \overline{z}_-| \\ 1 & \text{otherwise.} \end{cases} \tag{15}$$

where $\overline{z}_+$ and $\overline{z}_-$ are respectively the average state values of the corresponding actions $y = 1$ and $y = 0$ in the training data, and are computed by:

$$\overline{z}_- = \frac{\sum_{t=1}^{T} \sum_{i=1}^{N} z_i^t I(y_i^t = 0)}{\sum_{t=1}^{T} \sum_{i=1}^{N} I(y_i^t = 0)} \tag{16}$$

$$\overline{z}_+ = \frac{\sum_{t=1}^{T} \sum_{i=1}^{N} z_i^t I(y_i^t = 1)}{\sum_{t=1}^{T} \sum_{i=1}^{N} I(y_i^t = 1)} \tag{17}$$

where I is the indicator function.

4.3 Distributed NTT-FGM Learning

As a social network may contain millions of users and hundreds of millions of social ties between users, it is impractical to learn a NTT-FGM from a huge data using a single machine. Specifically, there are two major problems in our NTT-FGM model, namely, memory space and computing time. We use a sparse representation to solve the first problem. To speed up the computing, we deploy the learning task on a distributed system based on the MPI (Message Passing Interface).

MPI is a message-passing library interface specification. In the message-passing parallel programming model, data is moved from the address space of one process to that of another process through cooperative operations on each process. Based on the message passing scheme, we employ the *master-slave* model. That is, master can assign tasks to the slaves (computers), and combine the results in the master machine.

Specifically, in our learning algorithm, the time-consuming step lies in the calculation of the gradients, $\nabla_{\log \alpha_k}$, $\nabla_{\log \beta_{ij}}$, $\nabla_{\log \alpha_{ij}}$, which requires computing the inverse of the matrix A. Note A is a $NT \times NT$ matrix, which is too large to be held in memory when deal with a large data. Thus, we compute each column of A^{-1} respectively by solving the following linear equation

$$\forall\, i \;\; Ax_i = b_i \tag{18}$$

where x_i represents the i column of A^{-1} and b_i represents a NT-dimension vector, with the ith element 1, the other elements 0. Thus in each iteration, the master broadcasts the parameters to each slave and assigns the tasks to solve Equation (18) to the slaves averagely. All the salve computers calculate A, and send the results back to the master. The master reduces all the distributed results, and broadcasts the updated parameters to the slaves again for the next iteration.

5. EXPERIMENTAL RESULTS

The proposed approach for social action prediction is very general and can be applied to analyze different kinds of social networks. In this section, we present various experiments to evaluate the effectiveness and efficiency of the proposed approach. All data sets and codes are publicly available.[4]

5.1 Experimental Setup

Data Sets We perform our experiments on three different genres of real-world data sets: Twitter (a microblogging data set crawled from twitter.com), Flickr (a data set of photo sharing from flickr.com), and Arnetminer (a publication data set arnetminer.org).

- Twitter. The data set is crawled from Twitter by starting from the user "Carel Pedre (carelpedre)",[5] one of Haitian most popular radio DJs, who used Twitter to inform the world about the earthquake which ravaged his country. We extract all followers ($> 11,704$) of "carelpedre" and the users he is

[4] http://arnetminer.org/stnt/
[5] http://www.carelpedre.com/

following, and continue the process for each extracted Twitter user. We further crawl all tweets posted by the users as attributes. Finally, a data set used for action prediction consists of 7,521 users, 304,275 time varying following and followed relationships, and 730,568 tweets (blogs) posted by the users. A larger data set consisting of millions of users is also publicly available[4].

- Flickr. The data set is collected by [5], which contains 8,721 users, 485,253 friendships between users, and 2,504,849 favorite photos .

- Arnetminer. It is collected from ArnetMiner [30] and consists of 640,134 researchers, 1,554,643 coauthor relationship, and 2,329,760 publication papers by the researchers.

The action in Twitter is defined as the topic (e.g., "Haiti Earthquake") discussed by the user. More specifically, we selected several very relevant keywords, e.g., "Haiti", "earthquake", and "rescue". If a user posts a tweet containing the topic (keyword), we say that the user performs the action. We crawled the data from January 12nd, when the Haiti Earthquake struck, to January 26th. In the twitter data, we view one day as a time stamp. For example, a user called for donation for Haiti, his friends may respond by re-tweeting it, or posting a supporting message.

While the action of the Flickr data is defined as whether a user adds a photo to his favorite list. For example, if a user added a photo to his favorite list, his friends may also add the photo to their favorite. We extract the historic action log from 11/01/2006 to 03/20/2007 in the data set, dividing into 14 time stamps, 10 days a stamp.

The action of the Arnetminer data is defined as whether a researcher publishes a paper at a specific venue. For example, if a researcher published a paper at KDD, which may influence his collaborators to publish papers at KDD as well. The data is split into 10 time stamps, one for each year.

On all the three data sets, the attributes X is defined as the contents of tweets, information of photos, or related publication venues of the researcher. The content of each tweet is preprocessed by (a) removing stop-words and numbers; (b) removing words that appear less than three times in the corpus; and (c) lowercasing the obtained words. Then for each user, we combine all words in the remaining words in the tweets posted by the user and create the attribute vector by taking words as features.

Comparison Methods We compare the following methods for social action tracking:

SVM: It uses users' associated attributes as well as their neighbors' states to train a classification model and then employs the classification model to predict users' actions. For SVM, we employ SVM-light.[6]

wvRN: It employs a weighted-vote relational neighbor classifier [20] to train a classification model by making use of network information. In prediction, the relational classifier estimates the action state of a user by the weighted mean of his neighbors.

NTT-FGM: it uses the proposed NTT-FGM model to train the action tracking model and further uses the learned model for prediction.

According to our preliminary experiments, the σ in the Gaussian distribution does not significantly influence the performance. Thus, for simplicity, we empirically set $\sigma = 1$.

Evaluation Measures To evaluate our method, we consider the following three angles :

[6]http://svmlight.joachims.org/

Table 1: Performance of action prediction with different approaches (%).

Data set	Method	Recall	Precision	F1-Measure
Twitter	SVM	10.41	16.71	13.85
	wvRN	0.45	7.89	0.86
	NTT-FGM	26.40	21.14	23.47
Flickr	SVM	34.48	45.05	39.06
	wvRN	60.02	48.81	53.84
	NTT-FGM	56.18	45.80	50.47
ArnetMiner	SVM	10.19	21.62	13.85
	wvRN	14.83	16.39	15.57
	NTT-FGM	31.14	44.28	36.57

- **Prediction.** We evaluate the proposed model in terms of Precision, Recall, and F1-Measure, and compare with the baseline methods to validate the effectiveness of the proposed model.

- **CPU time.** It is the execution elapsed time of the model learning. This shows the speedup of the parallel implementation.

- **Case study.** We use several case studies as the anecdotal evidence to further demonstrate the effectiveness of our method.

The basic learning algorithm is implemented using C++ and all experiments are performed on a server running Ubuntu 8.10 with a AMD Phenom(tm) 9650 Quad-Core Processor (2.3GHz) and 8GB memory. The distributed learning algorithm is implemented under the MPI parallel programming model[7]. We perform the distributed training on 5 computer nodes (20 CPU cores) with AMD processors (2.3GHz) and 40GB memory in total. We set the maximum number of iterations as 250 and the threshold for the change of α, β, and λ to $1e-3$.

5.2 Prediction Performance

On all the three data sets, we use the historic users' actions to train the action tracking model and use the learned model to predict the users' actions in the last time stamp.

Table 1 lists the prediction performance of the different approaches on the three data sets with the following observations:

Performance comparison Our method NTT-FGM consistently achieves better performance comparing to the baseline methods. In terms of F1-Measure, NTT-FGM can achieve a $+10\%$ improvement compared with the (SVM). At the same time, NTT-FGM gives robust results, while the performance of wvRN is very sensitive to the data characteristics, with the highest F1-Measure on the Flickr data and extremely low value in the Twitter data. This is because on Flickr the user's action of adding favorite photos is mainly influenced by her friends' actions and wvRN can be viewed as a simple influence model, which makes wvRN mostly predicts "1" on Flickr, but the Twitter network (about "Haiti earthquake") in our experiment is relatively sparse, as a result wvRN outputs all "0". While our approach shows robust and consistent performance on all the data sets, which is important for the extendability of the methods.

Factor contribution analysis NTT-FGM captures three factors: 1) influence, 2) correlation and 3) personal interests/attributes. Next

[7]http://www.mcs.anl.gov/research/projects/mpich2/

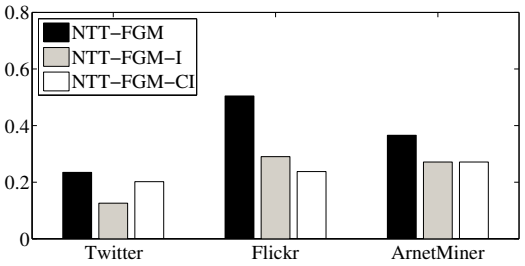


Figure 5: Contribution of different factor functions. NTT-FGM-I stands for our method by ignoring the influence factor function ($\lambda = 0$); and NTT-FGM-IC stands for NTT-FGM by ignoring both influence factor and correlation factor ($\lambda = 0, \beta = 0$).

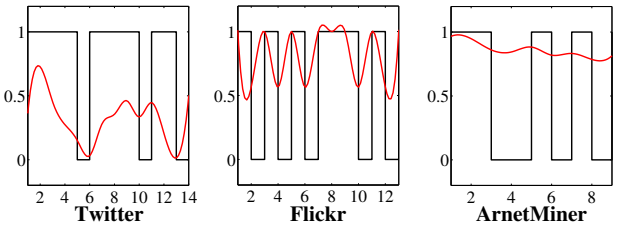


Figure 6: Example latent action states.

we perform an analysis to evaluate the contribution of different factors defined in our model. In particular, we remove those factors one by one (first influence factor function, followed by the correlation factor function), and then train and evaluate the prediction performance of NTT-FGM. Figure 5 shows the F1-Measure score after ignoring the factor functions. We can observe clear drop on the prediction performance, which indicates that our method works well by integrating the different factors for action tracking (prediction) and each defined factor in our method contributes improvement in the performance. Also, we find that the decrease varies on different data sets. On twitter there is a very low correlation between users' actions because users mainly post tweets on Twitter based on their previous experience or friends' tweets, and relatively act independently at a same time t.

Latent action states The learned latent action states essentially play a role as smoothing. Figure 6 illustrates several examples of the learned latent action states. It can be easily seen that the learned latent states (denoted as the red curve) is much more smoothing than the original discrete actions (denoted as the black step line), which indicates that latent action states can model the bias in binary actions. This is desirable for most prediction/classification tasks and further confirms us the advantage of the proposed NTT-FGM model.

5.3 Efficiency Performance

We now evaluate the efficiency of our approach by comparing the distributed learning algorithm with the basic one on the three data sets.

Table 2 lists the CPU time required for learning the NTT-FGM model on a single machine (Basic NTT-FGM) and by the distributed learning algorithm using 5 computer nodes (each 4 cores). The distributed learning algorithm typically achieves a significant reduction of the CPU time. For example, on Arnetminer, we obtain a speedup$> 17\times$, and on Flickr, the distributed learning algorithm results in a speedup$> 13\times$.

Table 2: Efficiency performance on the three data sets (5 computer nodes, each 4 cores)

Data Set	Basic NTT-FGM	Distributed NTT-FGM
Twitter	77.7hr	7.0hr
Flickr	9.14hr	0.68hr
Arnetminer	100min	6.2min

We also evaluate the speedup of the distributed learning algorithm using different numbers of computer nodes (5, 10, 15, 20 cores) to evaluate the cost of message passing. The speedup, as shown in Figure 7 (a), is close to the perfect line in the beginning. Although it decreases inevitably as the number of cores increases, it scales very well with $> 10\times$ speedup using 15 threads.

We further analyze how the network structure affects the efficiency of the learning algorithm. We generate a synthetic data set for this experiments by varying the density of the network ($\log \frac{|E|}{|V|}$). It can be seen from Figure 7 (b) and (c) that as the density (x-axis) increases, both basic learning and the distributed learning algorithm need more CPU time to train the NTT-FGM model, but the speedup of the distributed algorithm is consistently high (about $14 - 15\times$ using 20 threads).

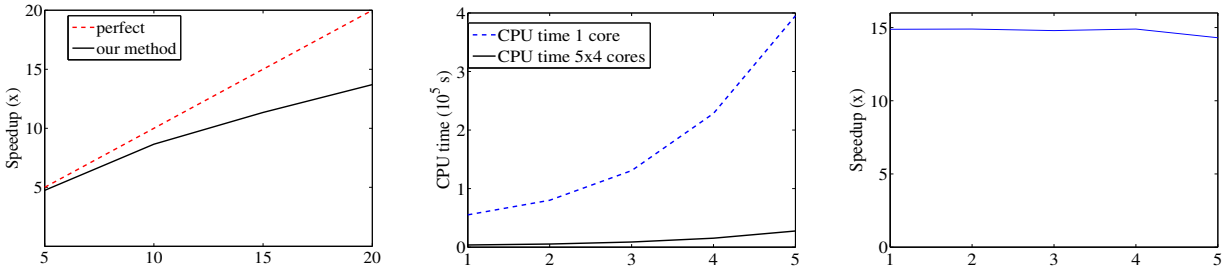
5.4 Qualitative Case Study

Now we present three case studies to demonstrate the effectiveness of the proposed model.

"Haiti Earthquake" The Haiti Earthquake is a devastating earthquake, leaving the country in shambles. We use our results to analyze people's actions related to the catastrophe on Twitter. Table 3 lists several example tweets about "Haiti Earthquake". We see that these tweets are about a call-for-donation by the famous tennis player "Serena Williams (serenajwilliams) ". The call-for message was soon retweeted by "actsofFaithblog" and "madameali" on their own microblogs, and a bit later the Haitian radio host "carelpedre" added a comment on Serena Williams's Twitter. These Twitter users are one of the most influential users and their actions on "Haiti Earthquake" quickly spread on Twitter with retweet and reply. (Because of this, Carel Pedre received a special "humanitarian" award at the second annual "Shorty Awards" in New York.) With the proposed model, we can identify the most influential users, whose actions can induce a large cascade followings, and track the information flows (via social ties with a high influence score or correlation score). In this way, we can understand how the influence spreads among people.

"Publication at KDD" We can also use the NTT-FGM model to track and predict who will publish (or submit) papers to KDD 2010. We train the NTT-FGM model using the Arnetminer data before 2009 and use the learned model to predict the latent action state of each researcher, and finally obtain a list of researchers ranked by the latent state. Table 4 lists a few representative examples selected from the top 100 ranked researchers. We see that our approach can not only find some famous researchers but also discover some "newcomers" to the KDD community. The first row lists several well-established researchers who have published a lot on KDD. The second row shows several "new" researchers who have no paper (or only few papers) published at KDD.

"Correlation between Researchers" Based on the learned NTT-FGM model, we can generate a correlation/influence map for better user analysis. Figure 8 shows an example correlation map between



(a)Speedup vs. #cores (b)CPU time vs. Network Density (c)Speedup vs. Network Density

Figure 7: Speedup Results. In graph (a), we evaluate the speedup varied with the number of cores. The x-axis stands for the number of cores, the y-axis represents the speedup($\frac{\text{CPU time 1 core}}{\text{CPU time } n \text{ cores}}$). In graph (b), we evaluate the CPU time with different network density(defined as $\log \frac{|E|}{|V|}$). The x-axis is $\log \frac{|E|}{|V|}$, the y-axis is the running time in seconds. In graph(c), we evaluate the speedup with different network density.The x-axis is $\log \frac{|E|}{|V|}$, the y-axis is the speedup.

Table 3: Action tracking on Twitter for "Haiti Earthquake".

Date/User	Tweet
6:03 PM Jan 16th by extratv	Tennis pro Roger Federer is joining forces with Rafael Nadal & @serenajwilliams to raise money for Haiti. http://su.pr/1E3MDU
5:23 AM Jan 17th by serena-jwilliams	Hey. Please, check out my foundation website: www.theswf.org to help those in Haiti!
6:48 AM Jan 17th by madameali	RT @SIXTWELVEMAG: RT @serenajwilliams: Hey. Please, check out my foundation website: www.theswf.org to help those in Haiti!
7:34 AM Jan 17th by actsof-Faithblog	RT @serenajwilliams: Hey. Please, check out my foundation website: www.theswf.org to help those in Haiti!
2:50 PM Jan 17th by carelpe-dre	@serenajwilliams Through Her 92k Mission has set a goal to contribute donations to the victims in #haiti. Visit www.theswf.org and donate

Table 4: Prediction on who will publish on (or submit to) KDD 2010. The examples are selected from the top 100 researchers predicted by the NTT-FGM model.

Frequent	Jiawei Han	Christos Faloutsos	Philip S. Yu
	Pedro Domingos	Lise Getoor	Jon M. Kleinberg
	Hang Li	ChengXiang Zhai	Wei-Ying Ma
	Lise Getoor	Jure Leskovec	Qiaozhu Mei
	Bing Liu	Jian Pei	Ravi Kumar
New	Huijia Zhu	Dimitrios Kotsakos	Zi Yang
	Noman Mohammed	Caimei Lu	Quanquan Gu
	Zhili Guo		

researchers. The strength of the link between two researchers indicates the correlation score. We see some researchers have strong correlation because they coauthored quite a few papers, e.g., Jiawei Han and Philip Yu. While our approach also finds some researchers have strong correlation, e.g., Ravi Kumar and Christos Faloutsos, although they only coauthored one or two paper(s). The discovered correlation can potentially benefit many applications such as link prediction. More correlation/influence analysis results can be found at http://arnetminer.org/stnt/.

6. RELATED WORK

Dynamic Social Network Analysis A number of models have been proposed to analyze dynamic social network with more and more dynamic information available in online social networks. Sarkar et al. [24] develop a generalized model associating each entity in Euclidean latent space and use kernel functions for similarity in latent space to model friendship drifting over time. Scripps et al. [25] present a model to investigate how different pre-processing decisions and different network forces such as selection and influence affect the modeling of dynamic networks. They also demonstrate the effects of attribute drifting and the importance of individual attributes in forming links over time. Yang et al. [33] propose a dynamic stochastic block model for finding communities and their evolutions in a dynamic social network. Zheleva et al. [35] propose a generative model which captures the statistical properties of these complex networks and the co-evolution of social and affiliation networks. Existing work on dynamic social network analysis mainly focuses on finding macro-level mechanisms of social dynamics, while our work tries to gain more insights into the micro-level dynamics of the social network.

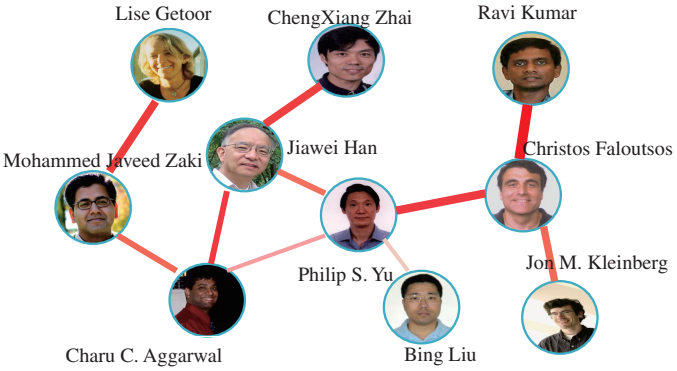


Figure 8: Example correlation analysis between researchers. The strength represents the correlation score between two researchers.

Social Influence Analysis Social influence analysis is an important research topic in social network analysis. One branch of social influence analysis is to verify the existence of social influence [2, 6, 7, 15, 23]. Anagnostopoulos et al. [2] focus on identifying and understanding social influence. They apply a statistical analysis method to identify and measure whether social influence is a source of correlation between the actions of individuals with social ties. Crandall et al. [6] have developed techniques for identifying and modeling the interactions between social influence and selection using data form online communities. Domingos and Richardson [7] investigate social influence in the customer network. They pro-

pose a model to identify customer's influence between each other in the customer network. They build a probabilistic model to mine the spread of influence for viral marketing [23]. A similar work is to maximize the spread of influence through a social network [15]. The basic idea is to find and convince a small number of influential users to adopt a product, and the goal is to trigger a large cascade for further adoptions through the effect of "word of mouth" (influence) in the social network. Another trend in social influence analysis is to quantitatively estimate the strength of the influence. Tang et al. [29] study the difference of the social influence on different topics and propose Topical Affinity Propagation (TAP) to model the topic-level social influence in social networks and develop a parallel model learning algorithm based on the map-reduce programming model. Goyal et al. [10] aims to learn the influence probabilities from historic users' actions. Compared with these social influence analysis works, we simultaneously model the social network structure, user attributes, and user actions into a unified model.

Group Behavior Analysis Group analysis is based on the view that deep lasting change can occur within a carefully formed group whose combined membership reflects the wider norms of society. There is an interest, in group analysis, on the relationship between the individual group member and the rest of the group resulting in a strengthening of both, and a better integration of the individual with his or her community, family and social network. Shi et al. [26] study the pattern of user participation behavior, and the feature factors that influence such behavior on different forum data sets. Tang et al. [31] employ relational learning to address the interdependency among data instances. Backstrom et al. [3] propose a partitioning on the data that selects for active communities of engaged individuals.

7. CONCLUSION

In this paper, we study a novel problem of social action tracking. We propose a noise tolerant time-varying factor graph model (NTT-FGM) to formalize this problem in a unified model. Three factor functions are defined to capture the intuitions discovered in our observation and an efficient algorithm is presented to learn the tracking model. A distributed learning algorithm has been implemented under the message-passing parallel programming model. We experiment on three different genres of data sets and further present a case study on social action prediction using the learned NTT-FGM model. Experimental results on three different types of data sets demonstrate that the proposed approach can effectively model the social actions and clearly outperforms several alternative methods for action prediction. The distributed learning algorithm also has a good scalability performance.

The general problem of social action tracking represents a new and interesting research direction in social network mining. There are many potential future directions of this work. A direct adaptation is to apply the NTT-FGM model for link prediction, which is important problem in social network. To add a user as a friend (e.g., follow somebody on Twitter) may be also influenced by social network structure, one's interests, and friends correlation. Another interesting issue is to extend the NTT-FGM model so that it can handle actions of multiple values. Another issue is to design the approach for (semi-)supervised learning. Sometimes, it may be difficult to collect sufficient labeled training data for an action. How to make use of the unlabeled data to help improve the prediction performance is an interesting problem.

8. *ACKNOWLEDGMENTS

Research was sponsored in part by National Natural Science Foundation of China (No. 60703059), Chinese National Key Foundation Research (No. 2007CB310803), and National High-tech R&D Program (No. 2009AA01Z138). Quan Lin is supported by NSFC (No. 70771043).

9. REFERENCES

[1] R. Albert and A. L. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1), 2002.

[2] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *KDD'08*, pages 7–15, 2008.

[3] L. Backstrom, R. Kumar, C. Marlow, J. Novak, and A. Tomkins. Preferential behavior in online groups. In *WSDM'08*, pages 117–128, 2008.

[4] N. T. Bailey. The mathematical theory of infectious diseases and its applications / norman t.j. bailey. 1975.

[5] M. Cha, A. Mislove, and K. P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *WWW'09*.

[6] D. Crandall, D. Cosley, D. Huttenlocher, J. Kleinberg, and S. Suri. Feedback effects between similarity and social influence in online communities. In *KDD'08*, pages 160–168, 2008.

[7] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD'01*, pages 57–66, 2001.

[8] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM'99*, pages 251–262, 1999.

[9] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine Learning*, 29(2-3):245–273, 1997.

[10] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *WSDM'10*, 2010.

[11] M. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1973.

[12] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *WWW'04*, pages 491–501, 2004.

[13] L. Guo, E. Tan, S. Chen, X. Zhang, and Y. E. Zhao. Analyzing patterns of user content generation in online social networks. In *KDD'09*, pages 369–378, 2009.

[14] S. S. Haykin. *Kalman Filtering and Neural Networks*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[15] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD'03*, pages 137–146, 2003.

[16] J. Kleinberg. Temporal dynamics of on-line information streams. In *Data Stream Managemnt: Processing High-speed Data*. Springer, 2005.

[17] D. Krackhardt. *The Strength of Strong ties: the importance of philos in networks and organization in Book of Nitin Nohria and Robert G. Eccles (Ed.), Networks and Organizations*. Cambridge, Harvard Business School Press, Hershey, USA, 1992.

[18] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML'01*, pages 282–289, 2001.

[19] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins.

Microscopic evolution of social networks. In *KDD'08*, pages 462–470, 2008.

[20] S. Macskassy and F. Provost. A simple relational classifier. In *Workshop on Multi-Relational Data Mining in conjunction with KDD'03*, 2003.

[21] M. E. J. Newman. The structure and function of complex networks. *SIAM Reviews*, 45, 2003.

[22] T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li. Multi-task feature learning. In *NIPS'08*, pages 1281–1288, 2008.

[23] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD'02*, pages 61–70, 2002.

[24] P. Sarkar and A. W. Moore. Dynamic social network analysis using latent space models. *SIGKDD Explor. Newsl.*, 7(2):31–40, 2005.

[25] J. Scripps, P.-N. Tan, and A.-H. Esfahanian. Measuring the effects of preprocessing decisions and network forces in dynamic network analysis. In *KDD'09*, pages 747–756, 2009.

[26] X. Shi, J. Zhu, R. Cai, and L. Zhang. User grouping behavior in online forums. In *KDD'09*, pages 777–786, 2009.

[27] P. Singla and M. Richardson. Yes, there is a correlation: - from social networks to personal behavior on the web. In *WWW'08*, pages 655–664, 2008.

[28] S. H. Strogatz. Exploring complex networks. *Nature*, 410:268–276, 2003.

[29] J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *KDD'09*, pages 807–816, 2009.

[30] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, pages 990–998, 2008.

[31] L. Tang and H. Liu. Relational learning via latent social dimensions. In *KDD'09*, pages 817–826, 2009.

[32] W. Wiegerinck. Variational approximations between mean field theory and the junction tree algorithm. In *UAI'00*, pages 626–633, 2000.

[33] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin. A bayesian approach toward finding communities and their evolutions in dynamic social networks. In *SDM'09*, pages 990–1001, 2009.

[34] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS'01*, pages 689–695, 2001.

[35] E. Zheleva, H. Sharara, and L. Getoor. Co-evolution of social and affiliation networks. In *KDD'09*, June 2009.

10. APPENDIX

10.1 Integration of Z

In this subsection, we introduce how we obtain the integration of Z. Equation 11 can be rewritten in the form of a multivariate Gaussian distribution. The standard formation of the integration of Multivariate Gaussian Distribution is as follows:

$$\frac{1}{(2\pi)^{\frac{m}{2}}|M|}\int_{\mathbf{u}}\exp\{-\frac{1}{2}(\mathbf{u}-\mu)^T M^{-1}(\mathbf{u}-\mu)\}d\mathbf{u}=1 \tag{19}$$

where $\mathbf{u}$ and μ is a m-dimension vector, M is a $m\times m$ matrix.

The idea here is to transform the exponential function $\exp\{.\}$ in Eq. (11) into a formation of multivariate Gaussian distribution.

$$\exp\{.\}\equiv\exp\{-\frac{1}{2}(\mathbf{z}-\mu)^T M^{-1}(\mathbf{z}-\mu)-c\} \tag{20}$$

where c is a value independent of $\mathbf{z}$. With further derivation, we can arrive

$$Z = \mathit{Const}\cdot|A|^{-\frac{1}{2}}\exp\{b^T A^{-1}b-c\} \tag{21}$$

where $\mathbf{b}=X\alpha$; $c=\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{d}\alpha_k x_{ik}^t$; A is a $NT\times NT$ block tridiagonal matrix, and $|A|$ is determinant of matrix A. The elements of A is defined as follows: (we use i^t to denote $i+(t-1)*N$ for simplicity.)

$$\begin{aligned}
A_{i^t,i^t} &= \sum_{k=1}^{d}\alpha_k+\sum_{j=1}^{N}\beta_{ij}m_{ij}^t+\sum_{j=1}^{N}\beta_{ji}m_{ji}^t+\sum_{j=1}^{N}\lambda_{ji}m_{ji}^{t-1}+\sum_{j=1}^{N}\lambda_{ij}m_{ij}^{t+1}\\
A_{i^t,j^t} &= A_{j^t,i^t}=-\beta_{ij}m_{ij}^t-\beta_{ji}m_{ji}^t\\
A_{i^t,j^{t-1}} &= A_{j^t,i^{t-1}}=-\lambda_{ji}m_{ji}^{t-1}-\lambda_{ij}m_{ij}^{t-1}
\end{aligned}$$

This construction matches our intuition. A_{i^t,i^t} represents the coefficient of $(z_i^t)^2$, while A_{i^t,j^t} represents the correlation factor, and $A_{i^t,j^{t-1}}$ describes the influence factor.

10.2 Model Learning Algorithm

The algorithm for model learning primarily consists of two steps. To summarize, in the first step, we fix $\mathbf{z}$ and update α,β,λ according to their gradients. We need to guarantee that $\alpha_k,\beta_{ij},\lambda_{ij}>0$. Thus, conventional gradient descent cannot be directly applied to the constrained problem. We employ a technique similar to that in [22]. Specifically we first maximize $\mathcal{O}(\theta)$ with respect to the log function. As a result, we get:

$$\begin{aligned}
\nabla_{\log\alpha_k} &= -\alpha_k(\sum_{t=1}^{T}\sum_{i=1}^{N}(z_i^t-x_{ik}^t)^2+\frac{\partial\log Z}{\partial\alpha_k})\\
\nabla_{\log\beta_{ij}} &= -\beta_{ij}(\sum_{t=1}^{T}(z_i^t-z_j^t)^2+\frac{\partial\log Z}{\partial\beta_{ij}})\\
\nabla_{\log\lambda_{ij}} &= -\lambda_{ij}(\sum_{t=1}^{T}m_{ji}^{t-1}(z_i^t-z_j^{t-1})^2+\frac{\partial\log Z}{\partial\lambda_{ij}})
\end{aligned} \tag{22}$$

where

$$\begin{aligned}
\frac{\partial\log Z}{\partial\alpha_k} &= -\frac{1}{2|A|}\frac{\partial|A|}{\partial\alpha_k}+\frac{\partial\vec{b}^T A^{-1}\vec{b}}{\partial\alpha_k}-\sum_{t=1}^{T}\sum_{i=1}^{N}{x_{ik}^t}^2\\
&= -\frac{1}{2}(A^{-T}):^T I:+X_{,k}^T A^{-1}\vec{b}-\vec{b}^T A^{-1}A^{-1}\vec{b}\\
&\quad+\vec{b}^T A^{-1}X_{,k}-\sum_{t=1}^{T}\sum_{i=1}^{N}{x_{ik}^t}^2\\
\frac{\partial\log Z}{\partial\beta_{ij}} &= -\frac{1}{2|A|}\frac{\partial|A|}{\partial\beta_{ij}}+\frac{\partial\vec{b}^T A^{-1}\vec{b}}{\partial\beta_{ij}}\\
&= -\frac{1}{2}(A^{-T}):^T\frac{\partial A}{\partial\beta_{ij}}:-\vec{b}^T A^{-1}\frac{\partial A}{\partial\beta_{ij}}A^{-1}\vec{b}\\
\frac{\partial\log Z}{\partial\lambda_{ij}} &= -\frac{1}{2|A|}\frac{\partial|A|}{\partial\lambda_{ij}}+\frac{\partial\vec{b}^T A^{-1}\vec{b}}{\partial\lambda_{ij}}\\
&= -\frac{1}{2}(A^{-T}):^T\frac{\partial A}{\partial\lambda_{ij}}:-\vec{b}^T A^{-1}\frac{\partial A}{\partial\lambda_{ij}}A^{-1}\vec{b}
\end{aligned} \tag{23}$$

where the notation $M:$ with a colon denotes the long column vector formed by concatenating the columns of matrix M.

In the second step, we fix α,β,λ to update $\vec{z}$, by solving a linear system:

$$(A+\mathbf{I})\vec{z}=\vec{y}+X\vec{\alpha} \tag{24}$$