

Like like alike — Joint Friendship and Interest Propagation in Social Networks

Shuang Hong Yang
Georgia Tech
shy@gatech.edu

Bo Long
Yahoo! Labs
bolong@yahoo-inc.com

Alex Smola
Yahoo! Research
smola@yahoo-inc.com

Narayanan Sadagopan
Yahoo! Labs
narayans@yahoo-inc.com

Zhaohui Zheng
Yahoo! Labs China
zhaohui@yahoo-inc.com

Hongyuan Zha
Georgia Tech
zha@cc.gatech.edu

ABSTRACT

Targeting interest to match a user with services (e.g. news, products, games, advertisements) and *predicting friendship* to build connections among users are two fundamental tasks for social network systems. In this paper, we show that the information contained in *interest networks* (i.e. user-service interactions) and *friendship networks* (i.e. user-user connections) is highly correlated and mutually helpful. We propose a framework that exploits homophily to establish an integrated network linking a user to interested services and connecting different users with common interests, upon which both friendship and interests could be efficiently propagated. The proposed *friendship-interest propagation* (FIP) framework devises a factor-based random walk model to explain friendship connections, and simultaneously it uses a coupled latent factor model to uncover interest interactions. We discuss the flexibility of the framework in the choices of loss objectives and regularization penalties and benchmark different variants on the Yahoo! Pulse social networking system. Experiments demonstrate that by coupling friendship with interest, FIP achieves much higher performance on both *interest targeting* and *friendship prediction* than systems using only one source of information.

Categories and Subject Descriptors

H.5.3 [Information systems]: Web-based Interaction; H.3.0 [Information search and retrieval]: Information filtering

General Terms

Algorithms, Performance, Experimentation

Keywords

Social network, link prediction, interest targeting

1. INTRODUCTION

Online social networking services have brought to the public a new style of social lives parallel to our day-to-day offline activities. Popular social network sites, such as Facebook, LinkedIn and Twitter have already gathered billions of extensively acting users and are still attracting thousands of

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0632-4/11/03.

enthusiastic newbies each day. Doubtlessly, social networks have become one of today’s major platforms for building *friendship* and sharing *interests*.

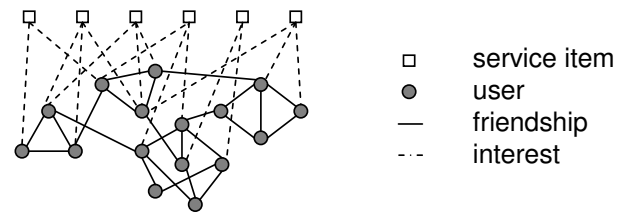


Figure 1: A social network graph. The connections consist of both (unipartite) edges within the user-user *friendship network* and bipartite user-item interactions in the *interest network*.

Fundamental to all social network services is the goal to effectively model the *interests* of a user and the *friendship* between users [21]. On the one hand, by capturing a user’s interests and accordingly exploiting the opportunity to serve her/him with potentially interesting service items (e.g. news, games, advertisements, products), one can improve the satisfaction of a user’s participation and boost the revenue of a social network site as well (e.g. via product purchases, virtual transactions, advertisement clicks). On the other hand, connecting people with common interests is not only important for improving existing users’ loyalty, but also helps to attract new costumers to boost the site’s traffic. In fact, *friendship prediction* (a.k.a. link prediction) and *interest targeting* (a.k.a. service recommendation) are two important tools available in almost all the major social network sites. Both activities which occur routinely in a social network have accrued a tremendous wealth of interaction traces, both *among users* (i.e. friendship network) and *between users and service items* (i.e. interest network). Figure 1 depicts a typical topology of a heterogeneous graph in the context of social networks.

1.1 Interests and Friendship

Modeling user interests and friendship in social networks raises unique challenges to both research and engineering communities. The information about a user’s behaviors is often scattered in both friendship and interest networks, involving other users that are closely connected to the user

and different activities that the user has engaged in. A fundamental mechanism that drives the dynamics of networks is the underlying social phenomenon of *homophily* [18]: people with similar interest tend to connect to each other and people of similar interest are more likely to be friends.

Traditional user profiling approaches often do not take full advantage of this fact. Instead they either employ feature engineering to generate hand-crafted meta-descriptors as fingerprint for a user [26, 5] or they extract a set of latent features by factorizing a user’s registered profile data; for example, by means of sparse coding [12] or latent Dirichlet allocation [2]. These approaches could be inaccurate because neither user friendship nor user behavior information is taken into account.

Recent approaches resort to collaborative filtering (CF) techniques [3, 23, 1, 10] to profile user interests by collaboratively uncovering user behaviors, where users are assumed to be unrelated to each other. While CF performs well in recommendation systems where decisions are mainly made individually and independently, it could fail in the context of social networks where user interactions substantially influence decision making [7, 18].

Modeling friendship is equally challenging. A typical social network is a graph both large and sparse, involving hundreds of millions of users with each being connected to only a tiny proportion of the whole virtual world. This property rules out traditional spectral algorithms for graph mining [19, 20] and calls for algorithms that are both efficient to handle large scale connections and capable of reliably learning from rare, noisy and largely missing observations. Unfortunately, progress on this topic to date is limited [13].

1.2 Friendship Interest Propagation

This paper exploits the important role homophily plays in social networks. We show that friendship and interest information is highly correlated (i.e. closely-connected friends tend to have similar interests) and mutually helpful (i.e. much higher performance for both friendship prediction and interest targeting could be achieved if coupling the two processes to exploit both sources of evidence simultaneously). We present a friendship-interest propagation (FIP) model that integrates the learning for interest targeting and friendship prediction into one single process.

The key idea in FIP is to associate latent factors with both users and items, and to define coupled models to encode both interest and friendship information. In particular, FIP defines a shared latent factor to assure dynamical interaction between friendship network and interest network during the learning process. In doing so, FIP integrates both interest and friendship networks to connect a user to both items of potential interest and other users with similar interests. FIP hereby provides a single unified framework to address both link prediction and interest targeting while enjoying the resources of both sources of evidence. Experiments on Yahoo! Pulse demonstrate that, by coupling friendship with interest, FIP achieves much higher performance on both tasks.

The contributions of this work are three-fold:

1. We present the friendship-interest propagation model that propagates two different types of evidence through heterogeneous connections.
2. We formulate the FIP model in a computational framework, discuss the flexibility in the choices of loss objectives (e.g. ℓ_2 , logistic regression, Huber’s loss) and reg-

ularization penalties (e.g. sparse coding, ℓ_2 penalties) and we benchmark different variants in a real-world social networking system;

3. For the implementation of FIP, we present a built-in scheme for bias correction based on pseudo-negative sampling to avoid overfitting, and we also deliver an optimization package that allows distributed optimization on streaming data.

Outline: §2 describes the background. §3 presents the detailed FIP model and our distributed implementation. §4 reports experiments and results. §5 reviews related work and §6 summarizes the results.

2. PROBLEM DEFINITION

We begin by briefly reviewing the state-of-the-art. This will come in handy as we will link them to our model in §3.

Modeling dyadic interactions is the heart of many web applications, including link prediction and interest targeting. Typically, a pair of instances from two parties (such as users and items), $i \in \mathcal{I}$ and $j \in \mathcal{J}$, interact with each other with a response $y_{ij} \in \mathcal{Y}$. The mapping

$$\{(i, j) \rightarrow y_{ij} \text{ where } i \in \mathcal{I}, j \in \mathcal{J}\}$$

constitutes a large matrix $Y \in \mathcal{Y}^{|\mathcal{I}| \times |\mathcal{J}|}$, of which only a tiny proportion of entries are observable; the goal is to infer the value of a missing entry $y_{\tilde{i}, \tilde{j}}$, given an incoming pair (\tilde{i}, \tilde{j}) . Essentially, the observed interactions define a graph, either unipartite (when $\mathcal{I} = \mathcal{J}$) or bipartite. The task amounts to propagating the sparse observations to the remainder (unobserved) part of the matrix. For convenience we will henceforth refer to i as *user* and j as *item* unless stated otherwise.

2.1 Interest Targeting

Interest targeting, or (service) recommendation, works with a bipartite graph between two different parties, e.g. user i and item j . It aims at matching the best item j^* to a given user i . We consider collaborative filtering (CF) approaches, which tackle the problem by learning from past interactions.

Neighborhood models. A popular approach to CF is based on the principle of *locality of dependencies*, which assumes that the interaction between user i and item j can be restored solely upon the observations of neighboring users or items [24, 17]. Such neighborhood-based models therefore propagate similar items to a particular user (item-oriented) or recommend a particular item to similar users (user-oriented). Basically, it predicts the interest of user i to item j by averaging the neighboring observations. For instance, the user-oriented model uses:

$$\hat{y}_{ij} = \frac{\sum_{i' \in \Omega_i} \omega_{ii'} y_{i'j}}{\sum_{i' \in \Omega_i} \omega_{ii'}}$$

where $\omega_{ii'}$ measures the similarity, e.g. Pearson correlation coefficient, between user i and its neighbor $i' \in \Omega_i$.

Latent factor models. This class of methods attempt to learn informative latent factors to uncover the dyadic interactions. The basic idea is to associate latent factors,¹

¹Throughout this paper, we assume each latent factor ϕ contains a constant component so as to absorb user/item-specific offset into latent factors.

$\phi_i \in \mathbb{R}^k$ for each user i and $\phi_j \in \mathbb{R}^k$ for each item j , and assume a multiplicative model for the interaction response

$$p(y_{ij}|i, j) = p(y_{ij}|\phi_i^\top \phi_j; \Theta).$$

This way the factors could explain past interactions and in turn make prediction for future ones. This model implicitly encodes the Aldous-Hoover theorem [6] for exchangeable matrices — y_{ij} are independent from each other given ϕ_i and ϕ_j . Parameter estimation for the model reduces to a low-rank approximation of the matrix Y that naturally embeds both users and items into a vector space in which the inner product $\phi_i^\top \phi_j$ directly reflect the semantic relatedness.

Latent factor models have gained tremendous successes in recommendation systems and have even become the current state-of-the-art for CF [10, 1]. A known drawback for such models is that, because it is learned only upon past interactions, the generalization performance is usually poor for completely new entities, i.e. unseen users or items, for which the observations are missing at the training stage. This scenario is well-known as the “cold-start problem” in recommendation systems. The recently proposed *regression based latent factor model* (RLFM) [1] addresses this problem by incorporating entity features into latent factor learning. The key idea is to use observable features to explain the learned latent variables (e.g. by regression or factorization). Suppose for each user and each item, there are observable features, x_i for i (e.g. user’s demographic information, self-crafted registration profiles) and x_j for j (e.g. content of a document, description of a product), as shown in Figure 2, RLFM [1] assumes the following dependencies:

$$\phi_i \sim p(\phi_i|x_i) \quad \phi_j \sim p(\phi_j|x_j) \quad y_{ij} \sim p(y_{ij}|\phi_i^\top \phi_j; \Theta).$$

Neighborhood based latent factor models. It is natural to combine the neighborhood models and latent factor models. A recent example is discussed [9], where the basic idea is to apply the *locality of dependencies* directly to the latent factors, for example:

$$\hat{\phi}_i = \frac{\sum_{i' \in \Omega_i} \omega_{ii'} \phi_{i'}}{\sum_{i' \in \Omega_i} \omega_{ii'}} \quad y_{ij} \sim p(y_{ij}|\hat{\phi}_i^\top \phi_j; \Theta). \quad (1)$$

This model² which is quite similar to [9] was deployed on the Netflix data yielding significantly better performances over both pure-neighborhood and pure latent factor models.

2.2 Friendship Prediction

Friendship (link) prediction recommends users to other users in the hope of acquainting people who were previously not connected in the network (or even unfamiliar with each other). Unlike interest targeting, the user network is unipartite. For a pair of users (i, i') the observation whether they are connected is a binary value $S_{ii'}$. Link prediction crucially influences both the traffic and the revenue of a social network and it is hence recognized as one of the key tasks in social network analysis.

Ideally, our goal is to learn a distribution over jointly exchangeable matrices (e.g. by applying the Aldous-Hoover factorization theorem). For reasons of practicality we pick a finite-dimensional factorization instead, which we shall discuss in the next section. Before we do so, let us briefly review existing approaches. Some of them employ random walk methods [14, 22] or spectral graph algorithms [19, 20].

²In this case the set of neighbors Ω_i contains i with $\omega_{ii} = 1$.

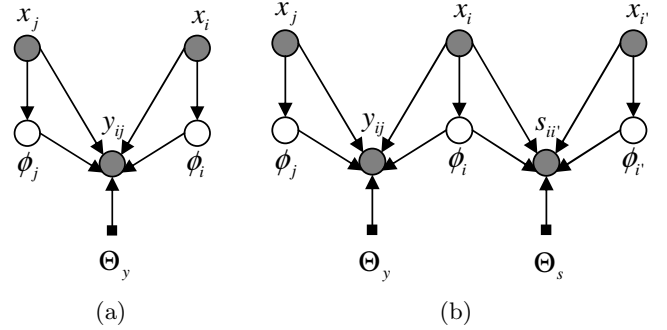


Figure 2: Graphical representations of (a) regression based latent factor model (RLFM) and (b) friendship-interest propagation model (FIP).

Random Walk. A random walk on the graph S is a reversible Markov chain on the vertexes \mathcal{I} . The transition probability from the vertex i to vertex i' is defined $p(i'|i) = s_{ii'}/d_i$. Here d_i denotes the degree of vertex i ; $s_{ii'}$ the connection weight between nodes i and i' . Vertexes are considered close whenever the hitting time is small or whenever the diffusion probability is large.

Spectral Algorithms. For the given network S , the unnormalized Laplacian is defined by $L = D - S$, where D is a diagonal matrix with $D_{ii} = d_i$. Spectral algorithms diffuse the connections by maximizing the spectral smoothness to obtain the intrinsic kinship defined by the dominant eigenvectors of the Laplacian

$$\sum_{i, i'} s_{ii'} \|u_i - u_{i'}\|^2 = 2ULU^\top, \text{ where } U = [u_1, \dots, u_{|\mathcal{I}|}]. \quad (2)$$

3. MODEL

We now consider interest targeting and link prediction in the context of social network, where evidence for both *interest* and *friendship* are available, allowing us to solve both tasks in a single framework. The rationale is that *friendship* and *interest* information are to some degree correlated,³ i.e. the network exhibits homophily [18] and the propagation of friendship and interest would be mutually reinforcing if modeled jointly.

In this section we present our model of friendship-interest propagation (FIP). We start with a probabilistic formulation, discuss different variants of the model and its implementation within an optimization framework, and then distinguish our model from existing works.

3.1 Probabilistic Model

The nontrivial correlation between interest and friendship motivates joint modeling of both sources of evidence. As shown in Figure 2, the friendship-interest propagation(FIP) model simultaneously encodes the two heterogeneous types of dyadic relationships: the user-item interactions $\{y_{ij}|i \in \mathcal{I}, j \in \mathcal{J}\}$, and user-user connections $\{s_{ii'}|i, i' \in \mathcal{I}\}$. Our model is built on latent factor models.

³Empirical analysis on Yahoo! Pulse illustrates that the interest correlation (Pearson score, max 1.0) between two directly-linked friends is 0.43, much higher than average.

Modeling Interest Evidence. To characterize the user-item dyads, y_{ij} , we assume that for each user i and item j there exist observable properties x_i (e.g. a user’s self-crafted registration files) and x_j (e.g. a textual description of a service item)⁴. Moreover, we also assume that there exist some subtle properties which cannot be observed directly, such as a user’s interests, a service item’s semantic topics. We denote these latent features by ϕ_i for i and ϕ_j for j respectively. We assume the response y_{ij} depends on both types of features (i.e. observable and latent):

$$\phi_i \sim p(\phi_i|x_i) \quad \phi_j \sim p(\phi_j|x_j) \quad y_{ij} \sim p(y_{ij}|\phi_i, \phi_j, x_i, x_j, \Theta),$$

where Θ denotes the set of hyper-parameters. To design a concrete model, one needs to specify distributions for the dependencies, $\phi_i|x_i$, $\phi_j|x_j$, and $y_{ij}|x_i, x_j, \phi_i, \phi_j$.

This model is essentially an integration of collaborative filtering [1] and content filtering [4]. On the one hand, if the user i or item j has no or merely non-informative observable features such that we have access to only their identity and past interactions, the model degrades to a factorization-style collaborative filtering algorithms [23]. On the other hand, if we assume that ϕ_i and ϕ_j are irrelevant, for instance, if i or j is totally new to the system such that there is no interaction involving either of them as in a cold-start setting, this model becomes the classical feature-based recommendation algorithms [3, 31, 4], which predict the interaction response y_{ij} purely based on the observed properties of i and j , and are commonly used in, e.g. webpage ranking [31], advertisement targeting [3], and content recommendation [4].

Modeling Friendship Evidence. We now extend the interest model to incorporate the social friendship-connection information among users. For this purpose, we define a random walk process for user-user networking. But unlike traditional random walk models [14, 22], we assume a user i is fully characterized by her observable features x_i and latent factor ϕ_i , and devise the following model for user-user transition:

$$\phi_i \sim p(\phi_i|x_i, \Theta) \text{ and } s_{ii'} \sim p(s_{ii'}|\phi_i, \phi_{i'}, x_i, x_{i'}, \Theta), \quad (3)$$

where $s_{ii'}$ reflects an observed state transition from i to i' . Unlike in random walk models where proximity in a graph is simply used to smooth secondary estimators of parameters (e.g. reachability, hitting times), we make direct use of it to model the latent variables ϕ_i . Note that whenever we restrict the norm of ϕ_i (e.g. by ℓ_2 regularization) and when we use an inner product model $\phi_i^\top \phi_{i'}$ to assess similarity, we approximately recover the graph Laplacian of Eqn.(2).

In this way our model integrates two different methodologies — collaborative filtering and random walks. It is different from traditional random walk models in which transition probability is defined solely based on graph topologies. It is also different from traditional CF models in that it is defined on unipartite dyadic relationships. By doing so, this integrated model not only allows learning of latent factors to capture graph topologies, but it also alleviates certain critical issues in random walks: for example, it naturally handles heterogeneous graphs (e.g. a compound graph consisting of both unipartite and bipartite connections such as Figure 1), and it also makes applicable computationally-efficient

⁴Whenever we do *not* have access to these properties we simply default to the expected value of the latent variables, which is easily achieved in a probabilistic model.

sequential learning algorithms (e.g. stochastic gradient descent), avoiding directly manipulating large matrices.

Friendship-Interest Propagation model. Based on the above descriptions, we finally summarize the overall FIP model in Figure 2 and the table below. Note that the tuples (i, x_i, ϕ_i) now play “double duty” in encoding interest interactions (i, j, y_{ij}) and friendship connections $(i, i', s_{ii'})$ simultaneously. Learning shared factors from coupled relationships gives us both more evidence and more constraints to work with, and in turn leads to better generalization.

| The Friendship-Interest Propagation (FIP) model. | |
|--|--|
| $\forall i \in \mathcal{I}$ | $\phi_i \sim p(\phi_i x_i, \Theta)$ |
| $\forall j \in \mathcal{J}$ | $\phi_j \sim p(\phi_j x_j, \Theta)$ |
| $\forall i \in \mathcal{I}, j \in \mathcal{J}$ | $y_{ij} \sim p(y_{ij} \phi_i, \phi_j, x_i, x_j, \Theta)$ |
| $\forall i, i' \in \mathcal{I}$ | $s_{ii'} \sim p(s_{ii'} \phi_i, \phi_{i'}, x_i, x_{i'}, \Theta)$ |

3.2 Model Specification

So far we deliberately described the FIP model in terms of general dependencies between random variables to make it explicit that the model is quite a bit more general than what can be achieved by an inner product model. Here, we specify the model within an optimization framework.

For computational convenience we assume linear dependencies between x_i and ϕ_i plus a noise term⁵ ϵ . This means

$$\phi_i = Ax_i + \epsilon_i \text{ where } \mathbf{E}[\epsilon_i] = 0. \quad (4)$$

$$\phi_j = Bx_j + \epsilon_j \text{ where } \mathbf{E}[\epsilon_j] = 0. \quad (5)$$

ϵ is typically assumed to be Gaussian or Laplace. Whenever nonlinearity in x is desired we can achieve this simply by using a feature map of x and an associated kernel expansion. Finally, we assume that the dyadic response (e.g. y_{ij}) depends on latent features only through the inner product (e.g. $\phi_i^\top \phi_j$) and on observable features through a bilinear product (e.g. $x_i^\top Wx_j$) [4]. That is:

$$y_{ij} \sim p(y_{ij}|f_{ij}) \text{ where } f_{ij} = \phi_i^\top \phi_j + x_i^\top Wx_j.$$

$$s_{ii'} \sim p(s_{ii'}|h_{ii'}) \text{ where } h_{ii'} = \phi_i^\top \phi_{i'} + x_i^\top Mx_{i'}.$$

Here, assume $x_i \in \mathbb{R}^m$ and $x_j \in \mathbb{R}^n$, the matrices $W \in \mathbb{R}^{m \times n}$ and $M \in \mathbb{R}^{m \times m}$ provide a bilinear form which captures the affinity between the observed features for the corresponding dyads. We also impose Laplace or Gaussian priors on W and M . One advantage of using an ℓ_1 (i.e. Laplace) prior is that it introduces sparsity, which makes (6) equivalent to sparse-coding [12] and thus improves both compactness and predictiveness of the learned latent factors ϕ .

Given observed responses for the dyads $\{(i, j) \in \mathcal{O}_y\}$ and $\{(i, i') \in \mathcal{O}_s\}$, the problem of minimizing the negative log-posterior of FIP boils down to the following objective:

$$\begin{aligned} \min \lambda_y \sum_{(i,j) \in \mathcal{O}_y} \ell(y_{ij}, f_{ij}) + \lambda_s \sum_{(i,i') \in \mathcal{O}_s} \ell(s_{ii'}, h_{ii'}) \\ + \lambda_{\mathcal{I}} \sum_{i \in \mathcal{I}} \gamma(\phi_i|x_i) + \lambda_{\mathcal{J}} \sum_{j \in \mathcal{J}} \gamma(\phi_j|x_j) \\ + \lambda_W \Omega[W] + \lambda_M \Omega[M] + \lambda_A \Omega[A] + \lambda_B \Omega[B], \end{aligned} \quad (6)$$

where λ ’s are trade-off parameters, $\ell(\cdot, \cdot)$ denotes a loss function for dyadic responses. The term $\gamma(\phi|x) = \Omega[\phi] +$

⁵Note that the latent ‘noise’ term is actually meaningful. It indicates the deviation of the user/item profiles from its cold-start estimates Ax_i and Bx_j respectively.

$\gamma_x(x, \phi)$. Here $\Omega[\cdot]$ is used to penalize the complexity (i.e. ℓ_2, ℓ_1 norm). The term $\gamma_x(x, \phi)$ regularizes ϕ by fitting the observed feature x , as defined by (6). This type of regularization are equivalent to applying content factorization (e.g. LSI, NMF, LDA) to the feature x in terms of a factor ϕ and bases A^{-1} or B^{-1} .

The motivations for a computational framework instead of direct probabilistic inference are mainly two-fold: First, the two formulations are somewhat equivalent — the distribution of the dyadic response (e.g. y_{ij}) and its dependence on the prediction (e.g. $p(y_{ij}|f_{ij})$) can be encoded precisely through the choice of loss functions; likewise, the prior over the observations or parameters could also be readily translated into the regularization penalties. Secondly, computational models allow more scalable algorithms, e.g. via stochastic gradient descent, whereas probabilistic reasoning often requires Monte Carlo sampling or quite nontrivial variational approximations.

3.3 Loss

In our case, both y and s are binary, i.e. $y_{ij}, s_{ii'} \in \{\pm 1\}$. We performed an extensive study in our experiments comparing a large variety of different loss functions. For the convenience of optimization, we limit ourselves to differentiable (in many cases, also convex) loss functions (see also Figure 3 for details):

Least Mean Squares: This is the most popularly-used loss in matrix factorization. It minimizes the Frobenius norm of the prediction residue matrix and leads to a SVD-style algorithm. We have the loss

$$\ell_2(y, f) = \frac{1}{2}(1 - yf)^2. \quad (7)$$

Lazy Least Mean Squares: This is a slight modification of ℓ_2 loss for the purpose of classification [30]. Basically, it is an iteratively truncated version of the ℓ_2 loss via

$$\mathcal{L}_2(y, f) = \min(1, \max(0, 1 - yf)^2). \quad (8)$$

It has been shown that this loss approximates the classification error rate in the example space [30].

Logistic regression: This is the loss used in a binary exponential families model. It is given by

$$\log(y, f) = \log[1 + \exp(-yf)]. \quad (9)$$

Huber loss: This is the one-sided variant of Huber’s robust loss function. It is convex and continuously differentiable via

$$\eta(y, f) = \begin{cases} \frac{1}{2} \max(0, 1 - yf)^2, & \text{if } yf > 0. \\ \frac{1}{2} - yf, & \text{otherwise.} \end{cases} \quad (10)$$

Ψ loss: Unlike other loss functions, which are all convex upper bound of the 0-1 loss, the Ψ loss [25] is non-convex. Both theoretical and empirical studies have shown appealing advantages of using non-convex loss over convex ones, such as higher generalization accuracy, better scalability, faster convergence to the Bayes limit [25, 30]. We implement the following version:

$$\Psi(y, f) = \begin{cases} \frac{1}{2} \max(0, 1 - yf)^2, & \text{if } yf > 0. \\ 1 - \frac{1}{2} \max(0, 1 + yf)^2, & \text{otherwise.} \end{cases} \quad (11)$$

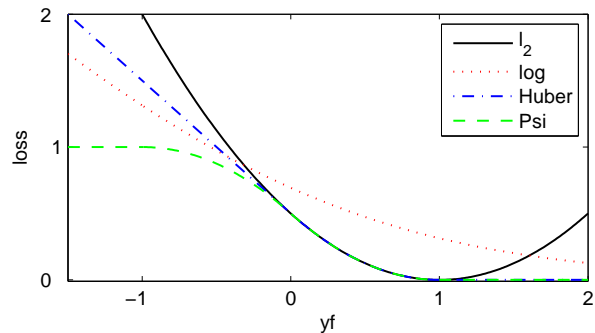


Figure 3: Least mean squares (ℓ_2), logistic (log), Huber and Ψ -loss (Psi). We use these four and the lazy ℓ_2 (omitted since its shape in parameter space is essentially identical to ℓ_2) loss for binary classification.

3.4 Bias Correction

A key challenge for learning latent factors from dyadic interactions is that the observations are extremely sparse with almost exclusively positive interactions observable. That is, we typically do *not* observe explicit information that user i does *not* like item j . Rather, the fact that we have not observed (i, j) suggests that i might not even know about j . In other words, absence of a preference statement or a social link should not be interpreted *absolutely* as negative information.

At the same time, unless we have access to negative signals, we will almost inevitably obtain an estimator that is overly optimistic with regard to preferences (e.g. predict positive for all the interactions). To balance both requirements we draw uniformly from the set of unobserved tuples (i, j) and (i, i') respectively and we require that, on average, observed pairs are preferred to unobserved ones.

In practice, since we use a stochastic gradient algorithm for minimization, for every positive observation, e.g. $y_{ij} = 1$, we randomly sample a handful set of missing (unobserved) entries $\{y_{ij'}\}_{j'=1:m}$, and treat them as negative examples (e.g. $y_{ij'} = -1$), with credibility $1/m$ each. Since the sampling procedure is random, the set of pseudo-negatives changes at each iteration and consequently each missing entry is treated as a potentially *very weak* negative instance.

3.5 Optimization and Implementation

Minimizing (6) is a nonconvex problem regardless of the choice of the loss functions and regularizers due to its use of bilinear terms. While there *are* convex reformulations for some settings, they tend to be computationally inefficient for large scale problems — the convex formulations require the manipulation of a full matrix which is impractical for anything beyond thousands of users. Moreover, the relationships between users change over time and it is desirable to have algorithms which process this information incrementally. This calls for learning algorithms that are efficient and amendable to dynamic updating so as to reflect upcoming data streams, rendering less attractive those offline learning algorithms such as classical SVD-based CF algorithms or spectral link prediction methods that involve manipulation of large-scale matrices. This requirement becomes more important for FIP than for traditional latent factor models

because we are now dealing with two (instead of one) large-scale coupled interactions and feature observations.

We established algorithms for distributed optimization based on the Hadoop MapReduce framework. The basic idea is to decompose the objective in (6) by optimizing with respect to y_{ij} and $s_{i'}$ independently in the Map phase, and to combine the results for ϕ_i in the Reduce phase.

Stochastic Gradient Descent. We briefly describe a stochastic gradient descent algorithm to solve the optimization of (6). The algorithm is computationally efficient and decouples different users. For a detailed discussion see [33].

The algorithm loops over all the observations and updates the parameters by moving in the direction defined by negative gradient. For example, for each $(i, j, y_{ij}) \in \mathcal{O}_y$:

$$\begin{aligned} - \phi_i &\leftarrow \phi_i - \delta \times (\ell'(y_{ij}, f_{ij})\phi_j + \lambda\Omega'[\phi_i]) \\ - \phi_j &\leftarrow \phi_j - \delta \times (\ell'(y_{ij}, f_{ij})\phi_i + \lambda\Omega'[\phi_j]) \\ - W &\leftarrow W - \delta \times (\ell'(y_{ij}, f_{ij})x_i x_j^\top + \lambda\Omega'[W]) \end{aligned}$$

To update on feature observations, for each $i \in \mathcal{I}$:

$$\begin{aligned} - \phi_i &\leftarrow \phi_i - \delta \times (\phi_i - Ax_i + \lambda\Omega'[\phi_i]) \\ - A &\leftarrow A - \delta \times ((Ax_i - \phi_i)x_i^\top + \lambda\Omega'[A]) \end{aligned}$$

where the subscripts of the trade-off parameters λ are omitted but clear from the context, δ is the learning rate⁶. Note that the gradient of ℓ_1 regularizer is the discontinuous sign function. We approximate it by a steep soft sign function: $\sigma(x) = \frac{1 - \exp(-\alpha x)}{1 + \exp(-\alpha x)}$, where α is a positive number controlling the ramp of $\sigma(x)$ (we use $\alpha = 100$).

Feature Hashing. A key challenge in learning FIP from large-scale data is that the storage of parameters as well as observable features requires a large amount of memory and a reverse index to map user IDs to memory locations. In particular in social networks with hundreds of millions of users the memory requirement would easily exceed what is available on today’s computers (100 million users with 100 latent feature dimensions each amounts to 40GB of RAM). We address this problem by implementing feature hashing [28] on the space of matrix elements. In particular, by allowing random collisions and applying hash mapping to the latent factors (i.e. ϕ), we make possible most-needed latent factors to remain in-memory, and in turn allow storing, accessing and updating (i.e. the stochastic gradient descent algorithm) to perform at sufficient speeds.

3.6 Discussion

We end this section with a brief discussion on how our model is related to the models discussed in §2.

Our first observation is that our FIP model indirectly induces a kernel for the friendship network graph: $k(i, i') = \phi_i^\top \phi_{i'}$ via the learned embedding ϕ . This is similar to the information diffusion kernel for graph [8, 11] in that both kernels inherent a Riemannian manifold for \mathcal{I} defined by the friendship network S , rather than a flat Euclidean space as in traditional CF models (e.g. neighborhood [24], factorization [23], RLFM [1]). However, it is also worth mentioning that the FIP induced kernel is different from diffusion kernels in that (i) our feature mapping ϕ_i is obtained from latent-factor based random walk model rather than topology-based random walk; and (ii) our model defines a compact low-rank manifold rather than a manifold that is potentially of infinite dimensionality [8, 11].

⁶We carry out an annealing procedure to discount δ by a factor of 0.9 after each iteration, as suggested by [9].

Traditional latent factor CF models [1, 23] work in Euclidean space where user factors ϕ_i are assumed identically and independently distributed: $\phi_i \sim \mathcal{N}(0, \sigma^2 I)$. Our model relates ϕ_i with one another by modeling the social network graph. This is equivalent to a row-correlated matrix-Gaussian $\Phi \sim \mathcal{MN}(0, \Sigma \otimes I)$, where $\Phi = [\phi_i, \dots, \phi_{|\mathcal{I}|}]^\top$, \mathcal{MN} is a matrix-variate Gaussian, and $\Sigma \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{I}|}$ defines the row-covariance (user-user covariance). By inexplicitly modeling the friendship manifold, our model hereby generalizes traditional latent factor CF models from Euclidean space to Riemannian space, in a way analogous to how diffusion kernels generalize Gaussian kernels.

Note that, although the neighborhood based latent factor model [9] also induces a manifold structure for \mathcal{I} , this manifold is virtual as it is directly constructed from Euclidean representations that essentially reflect the same amount of information. Our model generalizes this model by exploiting the true connections from social networks.

The FIP also differs from traditional link prediction algorithms. Actually, it borrows the idea of latent factor CF models to model the transition probability in terms of latent factors, making possible that (i) latent factors can be learned from network topologies; and (ii) connections can be propagated collaboratively through the interaction between latent factors. Essentially, our approach establishes an integrated network of interest and friendship that connects people with similar interests, and upon which both friendship and interests could be efficiently propagated.

4. EXPERIMENTS

We demonstrate the FIP model on Yahoo! Pulse in terms of both interest targeting and friendship prediction.

4.1 Yahoo! Pulse Data

Yahoo! Pulse (pulse.yahoo.com) is a social network site that allows users to create profiles, connect to friends, post updates, and respond to questions, as in other social networks. More importantly, it provides a mechanism for users to share interests, i.e. users can upload, download, install applications, and invite friends to try interesting applications. Our motivation is to utilize the user-user friendship network and user-application interest network from Yahoo! Pulse, so as to simultaneously propagate interest and friendship. We examine data collected on Yahoo! Pulse for about one year, involving hundreds of millions of users and a large collection of applications, such as games, sports, news feeds, finance, entertainment, travel, shopping, and local information services. Figure 4 shows the degree distribution of this data set. The data is very sparse and almost half of the users only have one friend connections and do not like any of the applications (they are essentially not using the network). Our goal is to propagate evidence to establish reliable connections both among users and between users and applications.

We use a subset of Yahoo! pulse data. The data set has 386 application items, 1.2M users, 6.1M friend connections and 29M interest interactions. There is a significant difference in the densities of the two networks in this data set. As the item set is pretty small, the interest network is relatively dense – each user likes 23.5 items on average. In contrast, as the user population is large, the friendship network is extremely sparse: on average, each user only has 4.9 friends out of the total 1.2 million.

Table 1: Service recommendation performance. We compared the following models: item oriented neighborhood model (SIM), regression based latent factor model (RLFM), neighborhood based latent factor model (NLFM), and friendship-interest propagation (FIP). For the latter we distinguish by choice of regularizer $\Omega[\cdot]$ and loss function ℓ as described in §3.3.

| Models | loss | $\Omega[\cdot]$ | AP@5 | AR@5 | nDCG@5 |
|--------|---------------|-----------------|--------------|--------------|--------------|
| SIM | | | 0.630 | 0.186 | 0.698 |
| RLFM | | | 0.729 | 0.211 | 0.737 |
| NLFM | | | 0.748 | 0.222 | 0.761 |
| FIP | ℓ_2 | ℓ_2 | 0.768 | 0.228 | 0.774 |
| FIP | lazy ℓ_2 | ℓ_2 | 0.781 | 0.232 | 0.790 |
| FIP | logistic | ℓ_2 | 0.781 | 0.232 | 0.793 |
| FIP | Huber | ℓ_2 | 0.781 | 0.232 | 0.794 |
| FIP | Ψ | ℓ_2 | 0.777 | 0.231 | 0.771 |
| FIP | ℓ_2 | ℓ_1 | 0.778 | 0.231 | 0.787 |
| FIP | lazy ℓ_2 | ℓ_1 | 0.780 | 0.231 | 0.791 |
| FIP | logistic | ℓ_1 | 0.779 | 0.231 | 0.792 |
| FIP | Huber | ℓ_1 | 0.786 | 0.233 | 0.797 |
| FIP | Ψ | ℓ_1 | 0.765 | 0.215 | 0.772 |

Table 2: Friendship prediction performance. We used the identical setting as in Table 1. The best results are printed in boldface.

| Models | loss | $\Omega[\cdot]$ | AP@5 | AR@5 | nDCG@5 |
|--------|---------------|-----------------|--------------|--------------|--------------|
| RLFM | | | 0.164 | 0.202 | 0.174 |
| FIP | ℓ_2 | ℓ_2 | 0.359 | 0.284 | 0.244 |
| FIP | lazy ℓ_2 | ℓ_2 | 0.193 | 0.269 | 0.200 |
| FIP | logistic | ℓ_2 | 0.174 | 0.220 | 0.189 |
| FIP | Huber | ℓ_2 | 0.210 | 0.234 | 0.215 |
| FIP | Ψ | ℓ_2 | 0.187 | 0.255 | 0.185 |
| FIP | ℓ_2 | ℓ_1 | 0.186 | 0.230 | 0.214 |
| FIP | lazy ℓ_2 | ℓ_1 | 0.180 | 0.223 | 0.194 |
| FIP | logistic | ℓ_1 | 0.183 | 0.217 | 0.189 |
| FIP | Huber | ℓ_1 | 0.188 | 0.222 | 0.200 |
| FIP | Ψ | ℓ_1 | 0.178 | 0.208 | 0.179 |

4.2 Evaluation Metrics

Both interest targeting and link prediction lead to a ranking of entities (e.g. items and users that the system may recommend) according to a score function. In our context this means that the scores f_{ij} and $h_{ii'}$ induce a ranking. Hence it is natural to use ranking metrics to assess performance. We consider the following three scores:

AP is the *average precision*. AP@ n averages the precision of the top- n ranked list of each query.

AR is the *average recall* of the top- n rank list of each query. **nDCG** or *normalized Discounted Cumulative Gain* is the normalized position-discounted precision score. It gives larger credit to top-ranked entities.

In all three metrics we use $n = 5$ since most social networks and recommendation sites use a similar number of items for friend and application suggestions; it is also the standard recommendations size used in the current system. For our evaluation we use cross-validation, where we randomly partition the data into two equally sized pieces and use one for

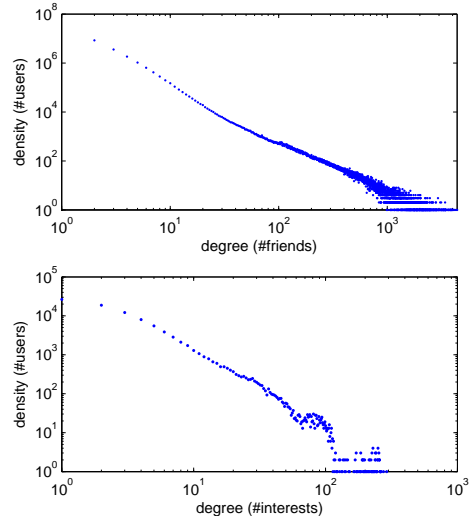


Figure 4: Degree distributions of Yahoo! Pulse friendship (top) and interest (bottom) networks.

training and the other for testing. All three measures are computed on testing data only, and they are averaged over five random repeats.

4.3 Interest Targeting

In this section, we report the results on interest targeting (i.e. application recommendation). We adopt a fairly strict evaluation by assessing the top results out of a total preference ordering of the item set for each user. In particular, for each user i , we consider all the 386 items as candidates; we evaluate the recommendation performance by assessing the quality of the top-5 items based on the comparison between ground truth (the actual list of the applications that user i liked) and the top-5 ranked shortlist outputted by each model.

For comparison, we take three popular CF models as baseline: the item-oriented neighborhood model (SIM), the regression based latent factor model (RLFM) [1], and the combination of them (referred to as neighborhood based latent factor model or NLFM [23]). SIM and RLFM use interest information; NLFM use both friendship information and interest information.

We test the baselines and different variants of FIP model, each of which is referred to in terms of the name combination of a loss and a regularization (e.g. FIP(ℓ_2, ℓ_2)). Table 1 demonstrate the overall results, i.e. the mean value of metrics averaged over 5 random runs. As the scale of the data is quite large, the predictive variance is very small (less than 0.002) and it is therefore not reported.

For the relatively dense interest network, all the reported models in our system actually achieve satisfactory performance in interest propagation. For most models, both the nDCG@5 and AP@5 scores are above 0.7, that is, out of the five recommended items, on average 3.5 are truly “relevant” (i.e. actually being liked by the user). Such performance is sufficiently satisfactory for propagating the 386 approved services in the current interest network. With such good performance, there is really not much room for further improvement. However, we still observe that noticeable improvements are obtained by the FIP models. Specifically,

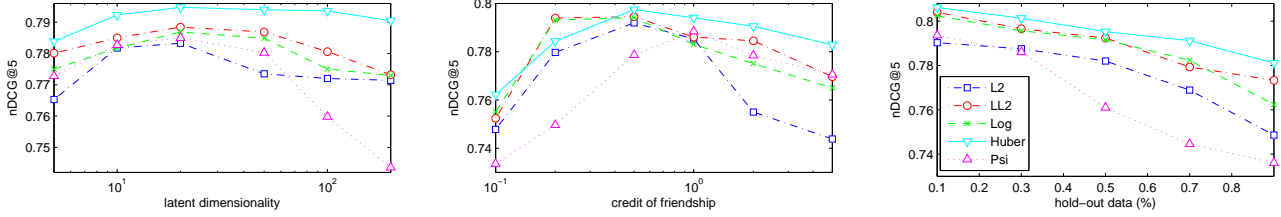


Figure 5: Service recommendation performance (nDCG@5) as a function of latent dimensionality (left), friendship credibility (mid) and the proportion of hold-out data (right).

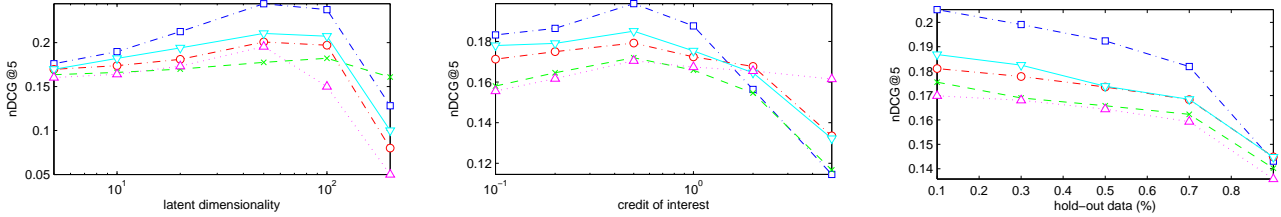


Figure 6: Friend prediction performance (nDCG@5) as a function of latent dimensionality (left), interest credibility (mid) and the proportion of hold-out data (right).

in terms of the nDCG@5 scores (similar comparisons apply to other metrics), FIP outperform the SIM model by up to 11.4%, RFLM by up to 10.8%, and NLFM by up to 4.7%. All improvements are significant (according to t -test with confidence threshold 0.01).

Among the 5 loss options for FIP, the one-sided Huber loss, the lazy ℓ_2 loss and logistic regression perform equally well (with Huber slightly better). Surprisingly, the nonconvex Ψ loss performs very poorly, even worse than ℓ_2 . We attribute this to the non-convexity of the Ψ loss – while non-convex losses perform superiorly in learning linear classifiers [25, 30], they could totally fail in learning the bilinear form of latent factor models because of the strong nonconvexity.

With respect to the two types of regularization, we observe that for each loss the ℓ_1 regularizer almost consistently outperforms ℓ_2 . As ℓ_1 regularization leads to compact (i.e. sparse) latent factors by assigning submissive latent dimensions to exactly 0, this observation suggests that sparseness can improve the informativeness of latent factors (being sparser implies smaller description length) and in turn leads to superior performance.

One of our claims is that friendship information is helpful for interest targeting. An interesting test would be to check how the credibility (i.e. λ_s) of the friendship influences the performance of interest prediction. We report this result in Figure 5. We can see that, as we increase λ_s , the performance first increases (it peaks at 0.5, or 1.0 for Ψ) and thereafter it starts to drop. This observation coincides with our intuitions: friendship information is truly useful for interest propagation, it helps interest targeting with discounted credit; yet, if too much weight is given to friendship, the latter may pollute the interest evidence and in turn harm interest targeting performance.

We also test the effects of two parameters: the dimensionality of latent factors k , and the proportion of hold-out testing data. Results are reported in Figure 5. For most losses, between 10 and 20 latent factors are sufficient for prediction. Also, with the exception of the Ψ loss, the performance is

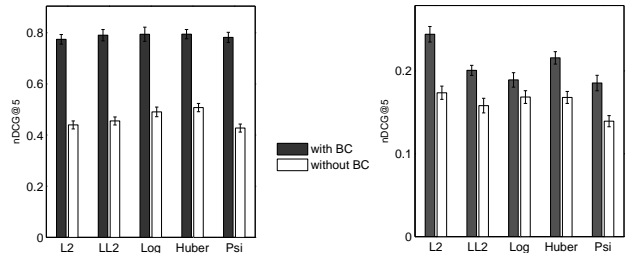


Figure 7: Recommendation performance in terms of nDCG@5 with and without bias-correction (BC) when applied to service recommendation (left) and friendship prediction (right).

quite stable to both parameters. This observation validates our hypothesis of the Ψ loss not being very amenable to efficient optimization: as latent dimension increases, more local optima are created and the Ψ loss performs worse; likewise, as training data becomes more sparse, the Ψ loss may be trapped in worse local optima.

An important procedure in our implementation is the bias-correction, i.e. generating pseudo-negative samples to correct the selection bias, as described in §3.4. We demonstrate the effects by comparing results obtained with and without this procedure in Figure 7. The comparisons are striking, indicating that latent factor models, if trained without negative examples, turn to overfit the training observations and misleadingly predict “positive” for most dyads. Our algorithms, by sampling missing interactions and using them as very weak negatives, guide the latent factor to capture the dyadic interactions while avoiding being fooled by the positive-only observations.

4.4 Friendship Prediction

We conducted similar evaluations on friendship propagation (i.e. link prediction). As the user population in the

Pulse social network is huge, it is prohibitive to take all the users as candidate friends and generate a total ordering of the whole user set for each user to evaluate the prediction performance; similarly, the models relying on neighborhood information (e.g. SIM and NLFM) are no longer tractable as they require computations quadratic in the number of users. To this end, we use a different evaluation mechanism: for each user i , we randomly sample M users that are not connected to i , we mix them with the set of users that i actually connects to. We then use this probe-polluted set as candidates, upon which the ranking performance is computed. In our experiments, we use $M = 300$ random probes per user.

We report the overall results in Table 2, where RLFM is used as the baseline model. The friend-network is extremely sparse (0.0039% density). Propagating friendship based on such sparse evidence is much more difficult, for example, RLFM only achieves 17% AP@5 and nDCG@5, which means out of the top-5 recommendations, less than 1 is truly relevant. Yet, we observe a significant improvement, as high as a 40% gain in nDCG@5, when FIP is used. This observation indicates that there is strong evidence of homophily in the Pulse social network such that users with similar interests are truly interested in each other. By leveraging the relatively dense interest evidence to assist the extremely sparse friendship graph, FIP achieves much higher performance in friendship predictions.

Regarding the loss functions, this time the ℓ_2 loss performs the best. We hypothesize that for much sparser friendship networks, losses that are more suitable for classification tasks tend to overfit the observed connections by making prematurely *hard* decisions to exclude connections that are not observed at the training stage. Similarly, because the ℓ_1 regularizer makes the latent factor sparser (some components of ϕ are shrunk to be exactly 0), it also turns to make *hard* predictions and in turn performs worse. Indeed, we observe significantly better performance for the ℓ_2 loss and/or regularizer, which turn to make smoother decisions.

The effects of parameter settings on this task are reported in Figure 6. We observe similar trends as in the previous task, although the performance is more sensitive: the performance changes faster as latent dimensionality increases or training data decreases. This matches the bias-variance analysis of statistic learning: as friendship connections are extremely sparse, we are typically dealing with a small-sample-size estimation, for which decrease of training data (e.g. increase hold-out proportion) or increase of model complexity (e.g. increase latent dimensionality) will inevitably lead to the increase of either bias or variance or both, and therefore the models are more likely to overfit the training observations and in turn generalize poorly.

As before, Figure 7 (right) shows that bias correction significantly improves the performance. Note that the difference is not as large as in interest targeting. This is likely due to the observation sparsity: in the sparser friendship network, two users that were not observed in the training set still have a good chance to be friends, which means many pseudo-negatives could be false-negatives.

5. RELATED WORKS

Collaborative filtering (CF) and link prediction were previously studied separately in two different research communities. The proposed FIP model bridges these two method-

ologies with a unified model. Essentially, FIP embeds all the users and items into the same space (e.g. Euclidean, simplex) so that the distances between two entities (e.g. user-item, user-user) reflect the *relatedness* (e.g. interest, friendship) between them, and hence, provides a unified treatment for both interest targeting and link prediction.

Existing approaches to link prediction diffuse the sparse connections using topology-based random walk [14, 22, 32] or spectral graph algorithms [19, 20], both of which involve expensive manipulation of large matrices. FIP borrows the idea of latent factor models in collaborative filtering [23, 1, 10] and it shows connections to random walk based models. As a side effect we obtain computationally attractive algorithms for efficient random walks.

Traditional CF techniques exploit past records of user behavior for future prediction based on either *neighborhood based* or *latent factor based* methods. The neighborhood latent factor model [9] merged these two models and reported significant performance improvement on the Netflix data. Though promising, the network structure exploited in this combined model [9] is a virtual one, constructed using the same evidence for learning latent factors. Our FIP model extends this model to allow the actual social network structure to be captured in latent factor learning.

Along another line, the recently proposed *regression based latent factor model* (RLFm) [1] incorporates node (user or item) features to improve recommendation performance in the *cold-start* scenario. The FIP model also generalizes RLFm [9] in a way analogous to how information diffusion kernels [8, 11] generalize the Gaussian kernels. Basically, instead of working in the Euclidean space as RLFm does, FIP induces a limited-dimensional Riemannian manifold defined by the topologies of both the unipartite friendship network and the bipartite user-service interaction network.

The FIP model has a close connection with recent works on collective matrix factorization [15, 29, 27, 32], where the tasks of learning relational data were also formulated in terms of factorizing multiple matrices. The current work continues our prior investigations on this topic and further examines interest and friendship propagation in the context of social networks, a task urgently motivated by emerging demands from social network services [21]. The techniques developed in this work also advances the state-of-the art from several aspects: (i) besides the dyadic relational data (i.e. edges), we also attempt to leverage the rich information conveyed by the node features using regression model similar to RLFm [1] or factorization models similar to sparse coding [12]; in this way, FIP integrates latent factor models [10, 23, 1] and predictive bilinear models [31, 4]; (ii) we present distributed optimization algorithms, address bias correction, discuss and benchmark different loss objectives and regularizers; and our work provides one of the first large-scale examinations of interest-friendship propagation in a real social network system.

One work relevant to ours is the social recommendation approach proposed by [16], where the trust relationships among users are used to improve cold-start recommendation. This model can be seen as a special case of our FIP model by assuming (i) no node feature x_i or x_j ; (ii) ℓ_2 loss objective; and (iii) asymmetric factor based random walk model, i.e. the transition probability is modeled as a multiplicative function of user-factor and a basis. Also, this work did not address the task of user relationships (e.g. trust,

friendship) propagation. Along this line, our work addresses both tasks with a more general framework and conducts large-scale evaluation on a social networking system.

6. CONCLUSIONS

Effectively modeling interest and friendship and accordingly recommending services and/or suggesting friends are fundamental to all social network services. In this paper, we have shown that the interest and friendship information is highly relevant and mutually helpful. We established a joint friendship-interest propagation model that leverages both evidences to address both tasks in one unified framework. The FIP model bridges collaborative filtering in recommendation systems and random walk in social network analysis with a coupled latent factor model. We conducted extensive experiments to benchmark different variants of FIP in the Yahoo! Pulse social networking system.

Two directions of future research appear attractive: The FIP model offers a latent factor for each user that captures both interest and friendship information. We plan to leverage such deeper user profiles to detect interest communities (i.e. grouping users according to interest with user-friendship in mind) and to identify the macro-behavior (i.e. the global effect as a result of individual actions) of each interest group. We also plan to investigate the underlying mechanism of how the interactions between users impact individual decision making in the context of social networks.

Acknowledgements

The authors would like to thank Su-Lin Wu, Yi Chang and the anonymous reviewers for helpful comments. Part of this work was done while Shuang-Hong Yang was on a summer internship at Yahoo! Labs. Part of the work of Shuang-Hong Yang and Hongyuan Zha was supported by NSF Grant IIS-1049694.

7. REFERENCES

- [1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD'09*.
- [2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.
- [3] Y. Chen, D. Pavlov, and J. F. Canny. Large-scale behavioral targeting. In *KDD'09*.
- [4] W. Chu and S.-T. Park. Personalized recommendation on dynamic content using predictive bilinear models. In *WWW'09*.
- [5] S. Gauch, M. Speretta, A. Chandramouli, and A. Micarelli. User profiles for personalized information access. In *The Adaptive Web*, 2007.
- [6] O. Kallenberg. *Probabilistic symmetries and invariance principles*. Springer, 2005.
- [7] T. Kameda, Y. Ohtsubo, and M. Takezawa. Centrality in sociocognitive networks and social influence: An illustration of group decision-making. *Journal Social Psychology*, 73(2):296–309, 1997.
- [8] R. I. Kondor and J. D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML'02*.
- [9] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD'08*.
- [10] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [11] J. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *JMLR*, 6:129–163, 2005.
- [12] H. Lee, R. Raina, A. Teichman, and A. Y. Ng. Exponential family sparse coding with applications to self-taught learning. In *IJCAI'09*.
- [13] J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *WWW'10*.
- [14] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM'03*.
- [15] B. Long, Z. M. Zhang, X. Wu, and P. S. Yu. Spectral clustering for multi-type relational data. In *ICML'06*.
- [16] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM'08*.
- [17] M. R. McLaughlin and J. L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *SIGIR'04*.
- [18] M. McPherson, L. S. Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.
- [19] M. E. J. Newman. Detecting community structure in networks. *European Physical Journal B*, 2004.
- [20] M. E. J. Newman. Modularity and community structure in networks. *PNAS*, 2006.
- [21] J. Owyang. The many challenges of social network sites. Web strategist blog, Feb.11, 2008.
- [22] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *PNAS*, 105(4):1118–1123, 2008.
- [23] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using MCMC. In *ICML'08*.
- [24] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW'01*.
- [25] X. Shen, G. C. Tseng, X. Zhang, and W. H. Wong. On psi-learning. *JASA*, 98:724–734, 2003.
- [26] M. Shmueli-Scheuer, H. Roitman, D. Carmel, Y. Mass, and D. Konopnicki. Extracting user profiles from large scale data. In *MDAC'10*.
- [27] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *KDD'08*.
- [28] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML'09*.
- [29] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. CofiRank—maximum margin matrix factorization for collaborative ranking. In *NIPS'07*.
- [30] S.-H. Yang and B.-G. Hu. A stagewise least square loss function for classification. In *SDM'08*.
- [31] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *NIPS'08*.
- [32] D. Zhou, S. Zhu, K. Yu, X. Song, B. L. Tseng, H. Zha, and C. L. Giles. Learning multiple graphs for document recommendations. In *WWW'08*.
- [33] M. Zinkevich, M. Weimer, A. Smola, and L. Li. Parallelized Stochastic Gradient Descent. In *NIPS' 10*.