Technical Report: Probabilistic Latent Relational Model for Integrating Heterogeneous Information for Recommendation

Yi Zhang , Jiazhong Nie
School of Engineering
University of California Santa Cruz
Santa Cruz, CA, USA
{yiz, niejiazhong}@soe.ucsc.edu

ABSTRACT

Many recommender systems might be part of an e-commerce or multi functional system (or portal) where various information about users, products/documents, social networks, and different types of user feedback about products/documents are available. This paper exploits the heterogeneous information a recommender system might collect to make the most appropriate recommendations. We propose a new Probabilistic Latent Relational Modeling (PLRM) approach to jointly model user, product, social network, and different types of implicit and explicit user feedback. A system can make different types of recommendations, such as who you may want to trust or what product you may want to buy, based on a single probabilistic model. Different types of user feedback collected can be utilized to improve the performance of recommendations. We also propose a hybrid variational Bayesian and Max A Posteriori method to estimate the parameters from various types of user feedback. The experimental results on two Epinion.com data sets demonstrate the effectiveness of the proposed modeling approach.

Categories and Subject Descriptors:
B.3.3 [Information Search and Retrieval]: Information filtering
General Terms:
Algorithms
Keywords:
recommender systems, information filtering, personalization, machine learning, relational models

1. INTRODUCTION

Recommender systems have achieved great success in industry applications. For example, online stores, such as *Amazon* and *Netflix*, provide customized recommendations for additional products or services based on a user's history. The latest offerings such as *My MSN, My Yahoo!*, *iGoogle* and Facebook's "like" button have attracted much attention due to their potential ability to infer users' interests from their behavior instead of requiring users to define their interest explicitly.

The major focus of recommender systems is to tailor the recommendation to each individual user. Many techniques have been proposed do so. For example, **Content-based adaptive filtering** studies the scenario where a recommendation system monitors a document stream and pushes documents that match a user profile to the corresponding user. The user may read the delivered documents and provide explicit relevance feedback, which the filtering system then uses to update the user's profile using relevance feedback retrieval models (e.g. Boolean models, vector space models, traditional probabilistic models, inference networks and language models) or machine learning algorithms (e.g. Support Vector Machines (SVM), K nearest neighbors (K-NN) clustering and logistic regressions. **Collaborative filtering (CF)** goes beyond merely using document content by leveraging information from other users with similar tastes and preferences to recommend items to a user . Memory-based heuristics and model based approaches have been used in collaborative filtering task [16, 3, 14, 13, 12, 26]. The top ranking algorithms in Netflix prize competition show that factorization based on models are very effective for CF [24, 20, 19, 25]. **Social network** analysis explores a user's social network to make recommendations [15, 21, 17, 8, 11]. For example, Kautz et. al. suggests combining social networks and collaborative filtering techniques for recommendation [15]. Golbeck shows that trust network is very useful for film recommendation [8]. Massa & Bhattacharjee showed the potential power of trust networks, and argued for a way of propagating trust over all users [22]. Konstas, Stathopoulos & Jose adopt the generic framework of Random Walk on social networks and found it further improves recommendation performance [17].

Over the last several years, research in standard recommender algorithms based on user-item rating matrix has been improved significantly. One current research focus is to go beyond the standard user-item rating matrix. How to use the rich and heterogeneous personal and contextual information to better model user preferences, describe items to be recommended, and make recommendation becomes an important problem. One can conclude from the literature that the following properties are desirable for a good system:

Using product meta data (product content): Product descriptions, such as the price of a product, the brand of a product, are used [1].

Using user meta data (user content): User features, such as the gender of a user, the location of the user, are used. Marketing researchers have developed various features, such as the Guadagni Little index [9], to characterize individual customers' purchasing preference.

Collaborative filtering (learning from others): Since we are learning a rich user model for each user, it may take a while before the system can gather enough data from the user and learn reliable model parameters. However, a good initial performance can be incentive for a new user to continue using the system. Borrowing information from other users can help alleviate this problem.

Using social networks: The social context of a user maybe helpful for inferring about the preferences of this user. For example, the 1,4046,851 "friends" of Barack Obama on Facebook [27] may be interested in news about the president's recent activities.

Learning from other feedback from the user: Besides explicitly rating an item, the system may collect other types of feedback from the user. For example, although many Amazon.com users do not provide much explicit feedback about products, they provide implicit feedback by purchasing a product, or clicking a page describing the product.

However, most of the existing literatures study each aspect individually. Some researchers have tried to combine two techniques together. For example, researchers have tried to combine contents of products with collaborative filtering and achieved performance better than each individual component[23, 2, 28, 1, 10]. Hao et al.[21] introduced a framework to combine social networks and collaborative filtering techniques, and they shown social network information helps recommender systems. Each aspect is addressing a specific part of the recommendation problem, and thus combining all of them may give the best possible performance. However, we are not aware of much prior work on providing a single unified solution that addresses all of the desired properties.

Building a personalized recommendation/search system that can integrate heterogeneous information and thus have all desirable property is the goal of this paper. Although improving each individual technique or desirable aspect is important, we believe the recommendation performance is limited by the information that particular technique or aspect could provide. This paper tries to provide a *unified modeling framework* to solve the problem. We propose to use a single probabilistic modeling approach to integrate different types of user feedback, user features, product contents, and social networks. We call the modeling framework Probabilistic Latent Relational model.

The general idea of our framework is to view users, documents, and items as objects. Each object may have meta data descriptions associated with it. Objects interact with (or relate to) each other, and the interactions correspond to various types of feedback in a recommender system. [1] Similar to factorization based collaborative filtering algorithms [20][19], each object has a hidden representation (vector) to be learned from the data. We model the a type of interaction/relation/feedback as a relationship function, and the inputs of the relationship function are the hidden vector representing the first object (such as the user who gives the feedback) and the vector representing the second object (the user/product that receives the feedback). The output of each relationship function is the value of the feedback. The vector representation of a user is decomposed into two parts. The first part is a hidden/latent representation of the user to be learned, and the second part is the feature vector derived from the user's meta data. Similarly, the hidden representation of an item is also decomposed into two parts.

We implement the proposed solution and evaluate it on an epinion.com data set we collected, where user meta data, user ratings, and social network information are available. The controlled experiments show that the proposed solutions works well in this scenario. We also evaluate it on a different epinion.com data set collected by other researchers, where only rating (or rating + trust network) information is available. We find our approach achieves better performance than those reported by other researchers on the same data set.

The organization of the remaining parts of this paper is as follows. Section 2 defines the problem space this paper tries to model. Section 3 describes the PLRM we proposed and an efficiently method to learn the model parameters. The experimental setting and results used to validate the proposed learning technique are reported in Sections 4 and 4.3. Section 5 summarizes and offers concluding remarks.

2. PROBLEM DEFINITION

The major goal of a recommender system is to recommend documents/item that are relevant to a user. As the system interacts with the user, it learns about the user from user feedback. The users can provide different types of feedback to the system. For examples, the user can rate an item (explicit feedback), purchase an item (implicit feedback), or identify a user as "trustable" (explicit feedback). Each piece of feedback provides the value of a relationship between two objects, where each object can either be a user or a product, depending on the feedback/relationship type.

The major research problem is how to make recommendation based on a variety of feedback, as well as meta data about products and users. To provide a unified solution, we first generalize the problem setting as follows. In this framework, we view everything involved (users, vendors, products, etc.) as objects. We represent each piece of feedback collected by the system as an asymmetric relational quadruplet (r, i, j, y), where r represents the type of relationship/feedback, i is the index of the first object, j is the index of second object of this relationship, and y is the value of the relationship/feedback. Without lose of generality, the rest of this paper uses $r = 1, 2, 3$ to represent the rating, purchasing and trust relationship/feedback respectively. $(1, i, j, 5)$

[1]Without lose of generality, we assume each relationship is between 2 objects. The approach can also be extended to handle relationships involving more than 2 objects.

means user i rates product j as 5. $(2, i, j, 0)$ means user i does not purchase product j, and $(3, i, j, 1)$ means user i trusts user j.

Besides the different types of feedback/relationships, the system also considers the features of the objects, e.g. the brand of a product, the loyalty index of a user for a brand, and the gender of a user. In the rest of this paper, we will use the following notation to represent the data the system deal with.

- N: the total number of unique objects
- i: the index of an object. $1 \leq i \leq N$.
- T: the total number of different types of objects ($T = 2$ in our experiments: user, product)
- R: the number of different relations (R=3 in our experiments: purchasing, rating, trust)
- r: the index of a relation type. $1 \leq r \leq R$.
- $t(i)$: the type of object i. $1 \leq t(i) \leq T$
- $\mathbf{f}_i$: the observed feature vector for object i.
- $D = \{(r, i, j, y)\}$: the training/labeled data set.
- $D' = \{(r, i, j)\}$: the testing data set.

The task is to predict the missing relationship value y for each triple in the testing data.

3. PROBABILISTIC LATENT RELATIONAL MODELS

Prior research has proposed different techniques to learn a user model from one or two particular feedback types. To make the most appropriate recommendation to the user, we propose to integrate all of the different types of feedback, including feedback on social networks, for recommendation. In particular, now we introduce the Probabilistic Latent Relational Modeling (PLRM) approach to achieve this goal. In PLRM, we introduce a hidden representation $\mathbf{h}_i$ for each object. For each relationship r, we introduce a hidden feature interaction matrix (A_r) and two observable feature transformation matrices ($W_{r,1}, W_{r,w}$) to capture how the feedback depends on the objects involved. The prediction for a relationship between two objects is given by a Gaussian distribution based on the corresponding representations, matrices, and features.

3.1 Model Definition

PLRM is a probabilistic graphical model. A special example of PLRM in the context of recommendation based on three different types of feedback (trust network, rating, and purchasing) is shown in Figure 1. The notations of the random variables in the model are described as follows:

- Hidden representation for object i: $\mathbf{h}_i$ (vector). $H = \{\mathbf{h}_i\}$
- Hidden feature transformation/interaction matrices for relation r: A_r (a matrix). $A = \{A_r\} = \{a_{r,*,*}\}$.
- Feature transformation/interaction matrices for relation r: $W_{r,1}$ (a matrix), $W_{r,2}$ (a matrix). $W = \{W_{r,1}, W_{r,2}\} = \{w_{r,l,*,*}\}, l = \{1, 2\}$. Matrix $W_{r,*}$ transforms the observed feature vector ($\mathbf{f}_i$) for object i to a lower dimensional vector $W_{r,*}\mathbf{f}_i$.

We assume the feedback value $y_{r,i,j}$ follows a Gaussian distribution:

$$\begin{aligned} y_{r,i,j} &\sim N(u_{r,i,j}, 1/\lambda_r^{(y)}) \\ u_{r,i,j} &= \mathbf{h}_i^T A_r \mathbf{h}_j + (W_{r,1}\mathbf{f}_i)^T W_{r,2}\mathbf{f}_j \\ &= [\mathbf{h}_i^T, \mathbf{f}_i^T][\begin{smallmatrix} A_r & 0 \\ 0 & W_{r,1}^T W_{r,2} \end{smallmatrix}][\begin{smallmatrix} \mathbf{h}_j \\ \mathbf{f}_j \end{smallmatrix}] \end{aligned}$$

where $u_{r,i,j}$ is the expected value of $y_{r,i,j}$, and $1/\lambda_r^{(y)}$ is the variance for the data of relation r. The center of the Gaussian, $u_{r,i,j}$, depends on the hidden representation of the objects involved ($\mathbf{h}_i$, $\mathbf{h}_j$), and the features (meta data) about the objects ($\mathbf{f}_i$, $\mathbf{f}_j$). When r is the rating relationship, the first part $\mathbf{h}_i^T A_r \mathbf{h}_j$ is the estimation based on users' long rating preferences, where A_r is a matrix modeling the rating interaction between user hidden representation and item hidden representation. The second term $(W_{r,1}\mathbf{f}_i)^T W_{r,2}\mathbf{f}_j$ captures the interaction between observable features.

$\{\mathbf{h}_i, A_r, W_{r,1}, W_{r,2}\}$ are the parameters of the model to be estimated from the training data set D. We assume the prior distribution of the parameters follow the Gaussian Distributions centered on 0:

$$\begin{aligned} \mathbf{h}_i &\sim N(\mathbf{0}, 1/\lambda^{(h)} I) \\ a_{r,*,*} &\sim N(0, 1/\lambda^{(A)}) \\ w_{r,l,*,*} &\sim N(0, 1/\lambda^{(w)}) \end{aligned}$$

$\lambda^{(h)}$, $\lambda^{(A)}$ and $\lambda^{(w)}$ are the variance of the above Gaussian distributions. They are the hyper parameters that characterize the shape of the prior distributions. The effect of the prior distribution is similar to the ridge regression (norm-2 regularizer) commonly used in machine learning algorithms for controlling model complexity and avoiding over fitting.

The proposed model is motivated by well performing recommendation models in the literature. It generalizes several existing models. If we model only one relationship, and set A_r to the zero matrix and $W_{r,1}$ to identify matrix, the model is similar to the bilinear model that works well on the Yahoo news recommendation task [6]. If we only model one rating relationship, and set A_r to the identify matrix and W_* to zero matrices, the model is similar to the well known norm-2 regularized singular value decomposition, which performs well on the Netflix competition [20][25].

Based on the above model assumptions, the joint likelihood of all random variables (H, A, W and D) in the system is:

$$P(H, A, W, D) = \prod_{(r,i,j,y) \in D} P(y_{r,i,j} | \mathbf{h}_i, \mathbf{h}_j, \mathbf{f}_i, \mathbf{f}_j, A_r, W_{r,1}, W_{r,w}, \lambda^{(y)}) \prod_i P(\mathbf{h}_i | \lambda^{(h)}) \prod_r P(A_r | \lambda^{(A)}) P(W_{r,1} | \lambda^{(w)}) P(W_{r,2} | \lambda^{(w)}) \quad (1)$$

Figure 1: A Probabilistic Latent Relational model. For simplicity, some random variables (A_r, $W_{r,*}$ $\lambda^{(*)}$ etc.) are not shown in this figure.

3.2 Parameter Estimation

"Over fitting" is a major challenge while estimating such a detailed user model, especially for users with limited data. Prior research has demonstrated that the Bayesian modeling approach helps alleviate the problem [20, 29]. A major principal of Bayesian method is to keep the distribution of parameters and average over the possible values of parameters (integral) while making a prediction. This is in contrast to the commonly used max a posterior (MAP) estimation or maximum likelihood (ML) estimation, where the most likely parameter (a point estimator) is selected and used in the final prediction. Prior collaborative filtering research has demonstrated that this full Bayesian modeling approach inference results in good performance on the Netflix movie recommendation task [20]. However, straightforward integration over the posterior estimation of all model parameters while doing Bayesian inference could be computationally intractable. Some techniques, such as Markov chain Monte Carlo and Laplace approximation, have been introduced to approximate the integral, among which Variational Bayesian seems a feasible solution in standard collaborative filtering settings [20].

Scalability and analytical tractability are also important issues that need to be addressed while taking the Bayesian approach to build a system like ours. For our particular task, the "over fitting" maybe a serious problem for user/item/object specific parameters $\mathbf{h}_i$ due to the limited data per user/item/object. This motivates us to follow the Variational Bayesian approach at the E step to estimate the distribution of the hidden representation of objects. However, it may not be a serious problem for user/item/object independent global parameters (A, W) given the large amount of data available to estimate these global parameters, assuming the recommender system is serving many users/items. Thus the posterior distribution of (A, W) is like to peak around the MAP point with a very small variance, and thus it is not worth the effort to do Variational Batesian on these parameters. Thus we chose to follow the MAP estimation using conjugate gradient descent method to get the MAP (a point estimator) of other parameters $(A_r, W_{r,1}, W_{r,2})$ at the M step.

The posterior distribution of $\mathbf{h}_i$ and MAP of $(A_r, W_{r,1}, W_{r,2})$ are combined to make the final predictions:

$$\hat{y}_{r,i,j} = \int_{\mathbf{h}_i, \mathbf{h}_j} P(\mathbf{h}_i)(\mathbf{h}_i^T A_r \mathbf{h}_j + (W_{r,1}\mathbf{f}_i)^T W_{r,2}\mathbf{f}_j) d\mathbf{h}_i \quad (2)$$

Now we derive the estimation of the distribution of $\mathbf{h}_i$ and the values of $(A_r, W_{r,1}, W_{r,2})$ using the following modified variational Bayesian EM algorithm.[2]

3.2.1 E Step:

In the E step, the Variational Bayesian approach is used to estimate the posterior distribution of H. Assuming (A, W) are known, based on Equation 1, we have:

$$P(H|A, W, D) \propto \prod_{(r,i,j,y)\in D} N(\mathbf{h}_i^T A_r \mathbf{h}_j + (W_{r,1}\mathbf{f}_i)^T W_{r,2}\mathbf{f}_j, 1/\lambda_r^{(y)}) \prod_{i=1}^{N} N(\mathbf{h}_i|\mathbf{0}, 1/\lambda^{(h)} I)$$

Using the above distribution for Bayesian inference in Equation 2 to predict y will result in intractable integrals. Thus we approximate the posterior by the following variational distribution:

$$Q(H) = \prod_{i=1}^{N} Q(\mathbf{h}_i)$$

The mean and covariance of these composite distributions are denoted as: $E_{Q(\mathbf{h}_i)}(\mathbf{h}_i) = \bar{\mathbf{h}}_\mathbf{i}$, and $Var_{Q(\mathbf{h}_i)}(\mathbf{h}_i) = \Sigma_i$

This variational distribution is restricted to belong to the Gaussian family so that replacing $P(H)$ with $Q(H)$ in Equation 2 will lead to straightforward prediction of $y_{r,i,j}$. With the intention that $Q(H)$ should be made very similar to the true posterior $P(H|A, W, D)$, we can estimate $Q(H)$ by minimizing the KL-divergence between it and $P(H|A, W, D)$. Since $Q(H)$ is factorized into individual $Q(\mathbf{h}_i)$, we can focus on one $Q(\mathbf{h}_i)$ at a time by fixing/ignoring other factors. With some derivation, we will have the following simple expectation over a quadratic form of h_i:

$$\begin{aligned} &E_{Q(H)}(\log Q(H) - \log P(H|A, W, D)) \\ &= E_{Q(\mathbf{h}_i)}(\frac{1}{2}\mathbf{h}_i^T \Sigma_i^{-1} \mathbf{h}_i - \mathbf{c}_i \mathbf{h}_i + \log Q(\mathbf{h}_i)) + Const. \quad (3) \end{aligned}$$

where

$$\begin{aligned} \Sigma_i^{-1} &= (\sum_{(r,i,j,y)\in D} \lambda_r^{(y)} A_r E_{Q(\mathbf{h}_i)}(\mathbf{h}_j \mathbf{h}_j^T) A_r^T \\ &\quad + \sum_{(r,j,i,y)\in D} \lambda_r^{(y)} A_r^T E_{Q(\mathbf{h}_i)}(\mathbf{h}_j \mathbf{h}_j^T) A_r + \lambda^{(h)} I) \\ &= (\sum_{(r,i,j,y)\in D} \lambda_r^{(y)} A_r (\bar{\mathbf{h}}_\mathbf{j} \bar{\mathbf{h}}_\mathbf{j}^T + \Sigma_j) A_r^T \\ &\quad + \sum_{(r,j,i,y)\in D} \lambda_r^{(y)} A_r^T (\bar{\mathbf{h}}_\mathbf{j} \bar{\mathbf{h}}_\mathbf{j}^T + \Sigma_j) A_r + \lambda^{(h)} I) \\ \mathbf{c}_i &= (\sum_{(r,i,j,y)\in D} \lambda_r^{(y)} \tilde{y} A_r \bar{\mathbf{h}}_\mathbf{j} + \sum_{(r,j,i,y)\in D} \lambda_r^{(y)} \tilde{y} A_r^T \bar{\mathbf{h}}_\mathbf{j}) \\ \tilde{y} &= y - (W_{r,1}\mathbf{f}_i)^T W_{r,2}\mathbf{f}_j. \end{aligned}$$

Based on Equation 3, we have $Q(\mathbf{h}_i) = N(\bar{\mathbf{h}}_i, \Sigma_i)$, where $\bar{\mathbf{h}}_i = \mathbf{c}_i \Sigma_i$.

3.2.2 M Step

In the M step, we estimate A and W based on the approximate posterior estimation of $Q(H)$ at E step. This can be done by maximizing the expected posterior likelihood of

[2]To make this paper more readable for the general ACM RecSys participants, we omit the detailed derivation in this paper and can add them to the appendix if needed.

A, W as follows:

$$\{\hat{A}, \hat{W}\} = \arg\max_{A,W} E_{Q(H)}(\log P(A, W, H|D)) = \arg\max_{A,W} E_{Q(H)} \log(P(D, A, W|H)) \quad (4)$$

Because $Q(H)$ is factorized into different $Q(\mathbf{h}_i)$, we can exam the expectation for each individual $\mathbf{h}_i$. With some operations, we can find the gradient of each $Q(h_i)$ factor with respect to each $A_r, W_{r,*}$ can be calculated as follows:

$$\frac{\partial E_{Q(\mathbf{h}_i)}(\log P(D, A, W|H))}{\partial A_r} = \sum_{(r,i,j,y)\in D} \lambda_r^{(y)} (\bar{\mathbf{h}}_i^T A_r \bar{\mathbf{h}}_j + (W_{r,1}\mathbf{f}_i)^T W_{r,2}\mathbf{f}_j - y) W_{r,2}\mathbf{f}_j \mathbf{f}_i^T + \lambda^{(w)} W_{r,1}$$

$$\frac{\partial E_{Q(\mathbf{h}_i)}(\log P(D, A, W|H))}{\partial W_{r,2}} = \sum_{(r,i,j,y)\in D} \lambda_r^{(y)} (\bar{\mathbf{h}}_i^T A_r \bar{\mathbf{h}}_j + (W_{r,1}\mathbf{f}_i)^T W_{r,2}\mathbf{f}_j - y) W_{r,1}\mathbf{f}_i \mathbf{f}_j^T + \lambda^{(w)} W_{r,2}$$

Based on the above gradient for each $Q(\mathbf{h}_i)$ factor and Equation 4, we can use the conjugate gradient descent method to find the MAP estimation of A, W at the M step.

4. EXPERIMENTS

4.1 Evaluation Data

We collected a data set from the web site Epinion.com. Epinions.com is a general consumer review site. The products are categorized into more than 20 categories (books, movies, electronics, etc.). Each review contains a particular user's rating about the product, and the rating is an integer from 1 to 5. Every user also maintains a "trust" user list, which provides the social network of trust relationships between users. The trust value is binary. The whole data set we collected contains 56,859 users, 271,365 different items, and 1,154,812 reviews/ratings. There are total of 603,686 trust statements among pairs of users. Most of the items are assigned into one category by Epinions. 10,994(19.3%) users only rate one item. The average number of ratings per user is only 20.3, and the average number of ratings per product is only 4.3. Compared to popular recommendation data set, such as Movielens and Netflix, this is a very challenging data set due to the sparsity.

Brand and *brand loyalty* have been identified by marketing researchers as very important characteristics about products and customers [9]. Thus we want to use brand related features for products and users. Although we expect this information could be available in real recommender systems, it is not explicitly provided by epinion.com, thus we extract it for a subset dataset. One subset contains products in the family/kids category, and it includes 19947 products, 14063 users and 105329 ratings in total. Each item has a 0/1 feature vector, where each dimension corresponds to a brand and the value is 1 if the item belongs to the corresponding brand, otherwise 0. Similarly, each user has a 0/1 brand feature vector and a dimension is 1 if the user has purchased any product in the corresponding brand, otherwise 0. Each user feature vector is a dynamic vector as the list of products purchased by the user changes over time.

We first split the data randomly into three parts: 80% as the candidates for training[3], 10% as the validation set to configure the system, and 10% as the testing set.

4.2 Evaluation Methodology

We design the experiments to answer the following questions:

1. Does the proposed framework work well in basic collaborative filtering settings.
2. Does integrating features, a variety of feedback the and social network improve the recommendation performance?

To answer the first question, we compare the performance of the proposed approach on a standard collaborative filtering setting ($PLRM_C$) where only the user-item rating matrix is available. The baseline algorithm we used is the probabilistic matrix factorization approach proposed by Salakhutdinov and Minh in [26], which works better than SVD and Netfix's own cinemax system on Netflix data set. We also evaluate the performance of the proposed approach on the task of using social network for recommendation ($PLRM_S$). The authors of [21] kindly shared their data set to us, and we use it in our first experiment. We control the experimental as described by [21], run our algorithm on their data set, and compare our performance with the results reported by their 2009 SIGIR paper. More details about this second data set is described in [21].

To answer the second question, we compared the performance with following settings:

R: Rating a model trained on user feedback on ratings

RP: Rating + Purchasing a model trained on user feedback on ratings and purchasing information of users.

RPT : Rating + Purchasing + Trust Network a model trained on ratings, purchasing information and the trust network between users.

RPTF: Rating + Purchasing+ Trust Network + Feature a model trained on ratings, purchasing information, the trust network and features.

The Epinion data set does not contain purchasing information. However, many other e-commerce web sites such as Amazon.com can collect more purchasing information than ratings. We convert the rating data into purchasing training data by changing all ratings to value 5, and we add in user and product pairs not rated into purchasing data as negative samples. Most e-commerce web sites have more purchasing information than ratings. To simulate the scenario of these

[3]Depending on the setting of each run, a varied portion of the 80% data are used for training.

Table 1: Comparison with existing approaches. Train ratio is the percentage of data with rating labels given. Dim is the number of hidden dimensions. The PMF and RSTE results are copied from Ma et.al.'s SIGIR09 paper.

Train raio @ Dim	PMF	RSTE	$PLRM_C$	$PLRM_S$
80% @ 5	1.183	1.135	1.034	1.033
90% @ 5	1.158	1.111	1.026	1.023
80% @ 10	1.176	1.126	1.034	1.032
90% @ 10	1.154	1.109	1.025	1.023

web sites, we only make a portion of the 80% rating training data available, and convert the rest into purchasing training data, assuming the users do not provide ratings on those items.

We focus on the product recommendation task and evaluate different approaches using two evaluation measure. The first one is to predict user rating of products, and we use the commonly used Root Mean Square Error $RMSE$ as the evaluation measure.

Although RMSE is commonly used in collaborative filtering research, we would like to validate our approach on a more realistic task. In the real-world, a major task for a recommender system is to recommend top items that the users may like. Hopefully the users will find some interesting item(s) to purchase. Minimizing RMSE does not necessarily optimizing the top recommendations. Thus we also simulate the ranking scenario and investigate the effect of our approaches on the ranking task. If we have complete relevance judgements from each individual user, we can carry out the ranking experiments in a standard information retrieval evaluation setting and report Precision and Recall measures. However, this is almost infeasible given the epinion.com data set. Thus we design our ranking experiments based on a variation of the evaluation method proposed by Koren et. al. [18]. For a user, each product rated as 5 by the user in the 10% testing set is considered a good/relevant testing point for that particular user. 1000 products randomly sampled are considered the irrelevant testing data points. For each user, the ratings of all the 1001 products are predicted using the trained model. All 1001 items are ranked according the score and the relative rank of the relevant product is recorded. This rank should be smaller for a better recommendation algorithm.

4.3 Experimental results

4.3.1 Comparing with existing algorithms

Table 1 shows that the proposed $PLRM_C$ and $PLRM_S$ models outperform two baseline models. This demonstrate that our approach works reasonably well in the standard filtering setting when only the user-item rating matrix is available (comparing $PLRM_C$ with PMF) or when trust network information is also available (comparing $PLRM_S$ with RSTE).

4.3.2 Comparing PLRMs with different information

Does integrating features, heterogeneous feedback and the social network improve the recommendation performance?

Table 2: RMSE comparison of different settings

training ratio	20%	40%	80%
R	1.06431	1.03427	0.997688
RP	1.06342	1.033224	0.996456
RPT	1.06215	1.0337	0.996155
RTPF	1.06129	1.03226	0.99533

Figure 2: Simulated ranking results with 80% training set

Table 2 compares the performance PLRM when different information is given. We found the RMSE is decreasing as more information is added. The improvement is bigger when the amount of training data (data with ratings) is smaller. One may ask whether this improvement is significant for a real recommender system and worth the effort, and we will address it in the next section.

4.3.3 Ranking results

Figure 2 compares the performance of PLRMs with different information given on the simulated ranking task. The x-axis is the cumulative rank distribution between 0% and 20% (top 200 ranked items out of 1001) and the y-axis is the portion of relevant products covered by this level of rank. Although the RMSE differences between R, RP, RPT and RPTF seem small (Table 2), the performance of RPTF is significantly better than other methods on the ranking task. We also observe similar results when the percentage of training data is smaller (20% or 40%). This is not surprising, as Koren has found that top-K recommendation task can better highlight the differences among methods than RMSE [18]. The results are very encouraging and show the PLRM approach has much potential in real ranking based recommendation tasks, and thus it worth to spend the extra effort on integrating various information. In particular, the brand information is very valuable for recommendations.

4.3.4 Similar Products

One useful function of a recommender system is to recommend similar products for a particular product, which we call a query. The Probabilistic Latent Relational Models can also provide this function, and the products similarity can be measured based on the hidden representations. However, the hidden representations are more reliable for items with a significant amount of feedback, and thus the similarity search results are reasonable in these cases. Some examples of similar products are shown in Table 3. For some products with few (1 or 2) ratings, the hidden representations learned are unreliable and thus the similar products found are not reasonable. In those cases, we may want to use the description/meta data to measure similarities. If a recommender system collects feedback on whether two items are similar, we can introduce a *similar* relationship between products into the PLRM framework for joint learning.

5. CONCLUSION

Although content based adaptive filtering, collaborative filtering, and social network have been researched for the recommendation task, the research is fragmented. Most existing techniques focus on using one or two kinds of user feed-

Table 3: Similar products. The number of ratings for each query is also provided.

	Name	ratings
Query	Little Smart Sort 'n Go Car	92
Neighbors	Little Smart Tiny Tot Driver	69
	Matchbox Special 5-Pack Vehicles	44
	Primo - Choo Choo Train	8
Query	Enfamil With Iron	202
Neighbors	Activity Walker	42
	Similac With Irons	34
	Munchkin White Hot Basic Spoons	22

back to learn a user model. As recommender systems become more popular and multi-functional, the problem of integrating a variety feedback types, meta data about users/items, and social networks for personalized recommendation is important and deserve research attention. We approach the problem by examining desirable characteristics of a recommender system and develop a new model, PLRM, as a unified probabilistic framework to build a such system. We generalize all user feedback types as different relationships between user and objects (user or item) that receive the feedback. The Probabilistic Latent Relational Model decomposes the model to be learned into two components: the hidden representations of users and items, and the probabilistic relational functions for different feedback types. The model is trained from various user feedback types. This enables the system to make recommendations to a particular user based on information from other users, the user's other types of feedback, the description of the user, the description of the products, the user's feedback on other products, and the user's social network. This modeling framework has more potential than many existing recommendation techniques because it tries to integrate heterogeneous information for recommendation.

This paper is a major step towards the direction of exploiting heterogenous feedback types, meta data and social networks for personalized recommendation. As the potential benefit of integrating heterogeneous information for recommendation are receiving more research attention and a new pioneering workshop on this topic will be held at ACM RecSys 2010 this year [4], we expect more research in this direction in the following years. We are also planning to continue this research and systematically propose several alternative information combination approaches and compare probabilistic latent relational models with them in the future. For the PLRMs, we leave much of the model refining tasks, such as choosing the number of hidden features, learning of the variances of the parameters, to future research. Other kinds of meta data, such as the location or the time, about users will be added. Our experiments consider only one network type, while the proposed solution can also be applied to situations when multiple different social networks co-exist. If we want to model real world social influence among users, the model should be modified. For mathematical convenience, we use Gaussian distributions in most of the places, while other functional forms can also be introduced in the future. Besides, we are also plan to go beyond two way interaction to model relationship between three or more objects, and a possible direction is to add tensor or polyadic factorization components into PLRMs [5].

Compared to popular factorization models for CF, the major difference of PLRM is to decompose the learning into two components: learning the representation of items ($\mathbf{h}_i$) and learning the relationships between items (A_r). This idea is motivated by recent research in neurolinguistics.

6. REFERENCES

[1] X. Bao, L. Bergman, and R. Thompson. Stacking recommendation engines with additional meta-features. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 109–116, New York, NY, USA, 2009. ACM.

[2] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

[3] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Technical report, Microsoft Research, One Microsoft Way, Redmond, WA 98052, 1998.

[4] P. Brusilovsky, I. Cantador, Y. Koren, T. Kuflik, and M. Weimer. International workshop on information heterogeneity and fusion in recommender systems (hetrec 2010). http://ir.ii.uam.es/hetrec2010/, 2010.

[5] Y. Chi, S. Zhu, Y. Gong, and Y. Zhang. Probabilistic polyadic factorization and its application to personalized recommendation. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 941–950, New York, NY, USA, 2008. ACM.

[6] W. Chu and S.-T. Park. Personalized recommendation on dynamic content using predictive bilinear models. In *WWW*, pages 691–700, 2009.

[7] L. Eliot. *How the brain and mind develop in the first five years of life*, chapter Language and the Developing Brain, pages 354–356. Bantam Books, 1999.

[8] J. Golbeck. Generating predictive movie recommendations from trust in social networks. In *Proceedings of the Fourth International Conference on Trust Management*, 2006.

[9] P. M. Guadagni and J. D. C. Little. A logit model of brand choice calibrated on scanner data. *Marketing Science*, 2(3):203–238, 1983.

[10] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 117–124, New York, NY, USA, 2009. ACM.

[11] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 53–60, New York, NY, USA, 2009. ACM.

[12] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference*, 1999.

[13] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 688–693, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[14] R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *Proceedings of the 27th annual international ACM SIGIR conference*, 2004.

[15] H. Kautz, B. Selman, and M. Shah. Referral web: combining social networks and collaborative filtering. *Commun. ACM*, 40(3):63–65, 1997.

[16] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[17] I. Konstas, V. Stathopoulos, and J. M. Jose. On social networks and collaborative recommendation. In *Proceedings of SIGIR-2009 ACM International Conference on Research and Development in Information Retrieval*. ACM Press, New York, US, 2009.

[18] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.

[19] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. In *IEEE Computer*, 2009.

[20] Y. J. Lim and Y. W. Teh. Variational Bayesian approach to movie rating prediction. In *Proceedings of KDD Cup and Workshop*, 2007.

[21] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of SIGIR-2009 ACM International Conference on Research and Development in Information Retrieval*, pages 203–210, 2009.

[22] P. Massa and B. Bhattacharjee. Using trust in recommender systems: An experimental analysis. In *Trust Management: Second International Conference, ITrust 2004, Oxford, UK, March/April 2004 Proceedings*, 2004.

[23] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, Edmonton, Canada, 2002.

[24] Netflix. Netflix prize. http://www.netflixprize.com (visited on Nov. 30, 2006), 2006.

[25] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *In: Proc. KDD Cup and Workshop*, 2007.

[26] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, 2008.

[27] TIME. *TIME Magazine*, chapter The World 10 Essentia Stories-Facebook's Favorite Politicians, page 10. July 2008.

[28] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *ACM SIGIR conference on Research and development in information retrieval*, 2006.

[29] Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 47–54, New York, NY, USA, 2007. ACM.