A Probabilistic Model for Collaborative Filtering with Implicit and Explicit Feedback Data

ThaiBinh Nguyen
Department of Informatics, SOKENDAI
Shonan Village, Hayama
Kanagawa, Japan 240-0193
binh@nii.ac.jp

Kenro Aihara
National Institute of Informatics
Hitotsubashi, Chiyoda
Tokyo, Japan 101-8430
kenro.aihara@nii.ac.jp

Atsuhiro Takasu
National Institute of Informatics
Hitotsubashi, Chiyoda
Tokyo, Japan 101-8430
takasu@nii.ac.jp

ABSTRACT
Collaborative filtering (CF) is one of the most efficient ways for recommender systems. Typically, CF-based algorithms analyze users’ preferences and items’ attributes using one of two types of feedback: explicit feedback (e.g., ratings given to item by users, like/dislike) or implicit feedback (e.g., clicks, views, purchases). Explicit feedback is reliable but is extremely sparse; whereas implicit feedback is abundant but is not reliable. To leverage the sparsity of explicit feedback, in this paper, we propose a model that efficiently combines explicit and implicit feedback in a unified model for rating prediction. This model is a combination of matrix factorization and item embedding, a similar concept with word-embedding in natural language processing. The experiments on three real-datasets (MovieLens IM, MovieLens 20M, and Bookcrossing) demonstrate that our method can efficiently predict ratings for items even if the ratings data is not available for them. The experimental results also show that our method outperforms competing methods on rating prediction task in general as well as for users and items which have few ratings.

KEYWORDS
Collaborative filtering, item embedding, implicit feedback, explicit feedback, matrix factorization

1 INTRODUCTION
Nowadays, recommender system (RS) has become a core component of many online services. RS analyzes users’ behavior and provides them with personalized recommendations for products or services that meet their needs. For example, Amazon recommends products to users based on their shopping histories; an online newspaper recommends articles to users based on what they have read.

Generally, an RS can be classified into two categories: Content-based approach and Collaborative Filtering-based (CF-based) approach. Content-based approach creates a description for each item and build a profile for each user’s preference. In other words, content-based approach recommends the items that are similar to items that interested the user. In contrast, CF-based approach [6, 7, 16, 19, 20] relies on the behavior of each user in the past, such as users’ ratings on items. CF-based approach is domain-independent and does not require content collection as well as content analysis. In this work, we focus on CF-based approach.

Basically, the data for a CF-based algorithm comes in the form of a rating matrix whose entries are observed ratings to items given by users. This kind of feedback is referred to as explicit feedback. Given these observed ratings, a typical task of CF-based algorithms is to predict the unseen ratings. One of the most efficient ways to perform CF is matrix factorization (MF) [6, 7, 19] which decompose the rating matrix into latent vectors that represent users’ preferences and items’ attributes. These latent vectors are then used to predict the unseen ratings. Usually, MF-based algorithms suffer from the sparseness of the rating matrix: if a user or an item has a very few number of ratings, it is difficult to find a “right” latent vector for the user or the item; in an extreme case, if rating data is not available for an item, MF-based algorithms can not find a latent vector for it.

When the ratings are not sufficient for modeling the items, exploiting other side information is a solution. Collaborative topic model [23] and content-based Poisson factorization [4] use text content information of items as a side information for recommending new items. Collaborative deep learning [24] models music content by deep neural network and combine with a MF-based model for music recommendation. However, in many cases, the items’ contents are not available or are not informative enough for modeling the items (e.g., when an item is described by a very short text, or only by some keywords).

In this work, instead of using such side information of items’ contents, we focus on utilizing another type of feedback, known as implicit feedback (e.g., clicks, views or purchase history), which can be easily collected with abundance during the interaction of users to the system. One way for recommending items that have no ratings is performing MF model based entirely on the implicit feedback only [5, 17, 18]. However, implicit feedback only is not reliable in capturing users’ preferences because it does not directly express the opinions of users to items. For example, a user’s click on an item does not mean that he/she likes the item; it may be the case that the user finally finds that he/she does not like it after clicking it. On the other hand, a user did not click an item may not be because he/she does not like item, it may be because he/she is unaware about the existence of it. Therefore, using implicit feedback only is not reliable for inferring the preferences of users or attributes of items.

To address the challenges above, we propose a probabilistic model that efficiently combines explicit feedback and implicit feedback in a unified model. We aim to make the model capable of modeling an item using both information from explicit feedback and implicit feedback data. For items that have many ratings, the representation of the item is mainly inferred from the rating data because it is more reliable. On the other hand, for items that has few or does
not have any ratings, the recommendation is mainly based on its implicit feedback data.

To find the representation for items without rating data, we exploit the implicit feedback with the assumption: if two items are usually clicked in the same context of each other, they are similar. This technique is referred to as item embedding which is very similar to word-embedding techniques [9, 11, 14] in natural language processing, which represent each by vectors that capture the relationships with its surrounding words.

In detail, our approach is a combination of two components: (i) the matrix factorization model [19] for finding representations of users and items based on explicit feedback, and (ii) the item embedding model for finding the representations of items that can capture the relationships among items based on implicit feedback. We will develop a probabilistic model that jointly learns the MF and item embedding model together. The item representations from item embedding model are learned to adapt with the item representations in MF model, and then can be used for rating prediction.

The rest of this paper is organized as follows. In Section 2 we formulate the problem and represent the background knowledge related to the method. Section 3 represents our idea in modeling items using implicit feedback and describes the probabilistic model for integrating implicit and explicit feedback data in a unified model. In Section 4, we present the effectiveness of our method by comparing with state-of-the-arts techniques using three public datasets. We show some related work in Section 5 and summarize the method as well as discuss some potential future directions in Section 6.

2 PRELIMINARY

2.1 Notation and Problem Formation

Let us establish some notations. We use \( u \) to denote a user and \( i \) or \( j \) to denote an item. Each observation of explicit feedback is represented by a triplet \((u, i, r_{ui})\) where \( r_{ui} \) is the rating that user \( u \) gave to item \( i \). The explicit feedback can be represented by a matrix \( R \in \mathbb{R}^{N \times M} \) where \( N \) is the number of users and \( M \) is the number of items. Each entry \( r_{ui} \) of matrix \( R \) is either the rating of item \( i \) given by user \( u \) or zero if the rating is not observed (missing entries). We use \( \mathcal{R} \) to denote the set of \((u, i)\)-pair that \( r_{ui} > 0 \), \( \mathcal{R}_0 \) to denote the set of item that user \( u \) gave ratings, and \( \mathcal{R}_I \) to denote the set of users that gave ratings to item \( i \).

The implicit feedback of a user-item pair is represented by a triplet \((u, i, p_{ui})\) where \( p_{ui} = 1 \) if implicit feedback of user \( u \) to item \( i \) is observed (i.e., the click, views or purchase of item \( i \) by user \( u \)), and \( p_{ui} = 0 \) if the implicit feedback is not observed. Implicit feedback is represented by matrix \( P \in \{0, 1\}^{N \times M} \).

Usually, explicit feedback matrix \( R \) is very sparse (i.e., with many missing entries). We are interested in predicting the missing entries of \( R \) based on the observed data.

2.2 Probabilistic Matrix Factorization

2.2.1 Basic probabilistic matrix factorization. Probabilistic matrix factorization (PMF) [19] is a method for modeling ratings which represents users and items by vectors in a shared latent space: user \( u \) is represented by vector \( \theta_u \in \mathbb{R}^d \) \((u = 1, 2, \ldots, N)\) and item \( i \) is represented by vector \( \beta_i \in \mathbb{R}^d \) \((i = 1, 2, \ldots, M)\), where \( d \) is the dimension of the latent space.

PMF assumes that the rating \( r_{ui} \) can be modeled by a normal distribution as follows.

\[
P(r_{ui}|\theta_u, \beta_i, \sigma_R^2) \propto N(\theta_u^T \beta_i, \sigma_R^2)
\]

where \( \theta_u \) and \( \beta_i \) are random variables that are drawn from multivariate Gaussian distributions:

\[
\begin{align*}
\theta_u & \sim N(0, \sigma_u^2 I_d) \\
\beta_i & \sim N(0, \sigma_i^2 I_d)
\end{align*}
\]

The parameters of the model \((\theta_{1:N}, \beta_{1:M})\) are learned by maximizing the log posterior distribution which is given in Eq.(3).

\[
\log P(\theta, \beta | R, \sigma^2) = -\frac{1}{2\sigma_R^2} \sum_{(u,i) \in \mathcal{R}} (r_{ui} - \theta_u^T \beta_i)^2 - \frac{1}{2\sigma_u^2} \sum_{u=1}^N ||\theta_u||^2 - \frac{1}{2\sigma_i^2} \sum_{i=1}^M ||\beta_i||^2 + C
\]

where \( \Theta = \theta_{1:N}, \beta = \beta_{1:M}, \sigma = (\sigma_R^2, \sigma_u^2, \sigma_i^2) \), and \( C \) is a constant when \( \sigma_R, \sigma_u, \sigma_i \), are fixed.

Maximizing Eq.(3) is equivalent to minimizing the following error function.

\[
L(\theta, \beta) = \frac{1}{2} \sum_{(u,i) \in \mathcal{R}} (r_{ui} - \theta_u^T \beta_i)^2 + \frac{\lambda_u}{2} \sum_{u=1}^N ||\theta_u||_F^2 + \frac{\lambda_i}{2} \sum_{i=1}^M ||\beta_i||_F^2
\]

where ||.||_F is the Frobenius norm of a vector, \( \lambda_u = \sigma_R^2/\sigma_u^2 \), \( \lambda_i = \sigma_R^2/\sigma_i^2 \). This optimization problem can be efficiently solved by either Alternating Least Square (ALS) method as described in [5] or by stochastic gradient descent (SGD).

2.2.2 PMF with biases. As analyzed in [8], much of the ratings due to effects associated with either users or items, which are independent of their interaction. A simple example is that some users tend to give higher ratings than others, and some items trend to receive higher ratings than others. Therefore, it is useful to introduce a bias term for each user and item. The rating model with bias terms is given in Eq. 5.

\[
P(r_{ui}|\theta_u, \beta_i, \mu, b_u, c_i, \sigma_R^2) \propto N(\mu + b_u + c_i + \theta_u^T \beta_i, \sigma_R^2)
\]

where \( \mu \) is the global mean of the ratings, \( b_u \) and \( c_i \) are the bias terms of user \( u \) and item \( i \) respectively. Bias terms are also random variables, and we also put a zero-mean Gaussian distribution prior on them as well:

\[
\begin{align*}
b_u & \sim N(0, \sigma_b^2) \\
c_i & \sim N(0, \sigma_c^2)
\end{align*}
\]

\( \theta_{1:N}, \beta_{1:M}, \mu_{1:N}, c_{1:M} \) are estimated by maximizing the log posterior distribution, which is equivalent to minimizing the following
error function.

\[ L(\Theta, \beta, b, c) = \frac{1}{2} \sum_{(u, i) \in R} [r_{ui} - (\mu + b_u + c_i + \theta_u^T \beta_i)]^2 \\
+ \frac{\lambda_a}{2} \sum_{i=1}^{N} ||\theta_u||_F^2 + \frac{\lambda_b}{2} \sum_{i=1}^{M} ||\beta_i||_F^2 \tag{7} \]

where \( b = b_{1:N}, c = c_{1:M}, \lambda_b = \sigma_R^2/\sigma_b^2 \), and \( \lambda_c = \sigma_R^2/\sigma_c^2 \).

3 PROPOSED METHOD

In this section, we describe our method, which is a combination of probabilistic matrix factorization for rating prediction and item embedding model for implicit feedback data. First, we will present the idea of item embedding model.

3.1 Item embedding model based on implicit feedback

Inspired by word embedding techniques [9, 11, 12, 14, 15] which represent a word by vectors that capture the relationship with its surrounding words, we apply the same idea to find representations of items based on implicit feedback data.

Similar to words, items also have their contexts, which is a model choice and can be defined in different ways. For example, the context can be defined as the set of items that are clicked by the user (user-based context); or can be defined as the items that are clicked in a session with a given item (session-based context). In this work, we use the user-based context, which is defined as follow: given item \( i \) that is clicked by user \( u \), the context of \( i \) is the list of all items that \( u \) have clicked.

The item embedding model presented in this section is partly based on the word-embedding model presented in [12] which we bring into the world of items with a change: in [12], each word is represented by a unique vector while in this item embedding model we use two vectors to represent each item. We found that, a model that uses two vectors for representing items can be efficiently trained by parallelizing the algorithm for the optimization problem (see our discussion on parameter learning in Section 3.2).

We present each item by two vectors: an item vector \( \rho_i \) and a context vector \( \alpha_i \). These two vectors have different roles: the item vector describes the distribution of the item, and the context vector describes the distribution of the co-occurrence of an item with other items in its context.

The model describes the appearance of an item conditional on other items in its context as follows.

\[ p(i|j) = f(i, j)p(i) \tag{8} \]

where \( p(i) \) is the probability that item \( i \) appears in the data, \( p(i|j) \) is the probability that \( i \) appears in context of \( j \) and \( f(i, j) \) is the link function that reflects the association between \( i \) and \( j \). The role of the link function is straightforward: if item \( i \) is often clicked (i.e., \( p(i) \) is high), however, \( i \) and \( j \) are not often clicked together (i.e., \( p(i|j) \) is low) then the link function \( f(i, j) \) should have small value. On the other hand, if \( i \) is rarely clicked (i.e., \( p(i) \) is low) but if \( i \) and \( j \) are often clicked together (i.e., \( p(i|j) \) is high), the link function \( f(i, j) \) should have high value.

There are different choices for the link functions which lead to different embedding models. Following the work in [12], we choose the link function \( f(i, j) = \exp(\rho_i^T \alpha_j) \). Combine with Eq.(8) we have: \( p(i|j) = \exp(\rho_i^T \alpha_j) \), or:

\[ \log \frac{p(i|j)}{p(i)} = \rho_i^T \alpha_j \tag{9} \]

Note that \( \log \frac{p(i|j)}{p(i)} = \log \frac{#(i, j)}{#(i)} \) is the point-wise mutual information (PMI) [3] of \( i \) and \( j \). Eq.(10) can be rewritten as follows.

\[ \rho_i^T \alpha_j = PMI(i, j) \tag{10} \]

Empirically, PMI can be estimated using the actual number of observations in the implicit feedback data.

\[ PMI(i, j) = \log \frac{#(i, j)}{#(i)\#(j)} \tag{11} \]

where \( D \) is the set of all item-item pairs that are observed in the click history of any user, \( #(i) \) is the number of times item \( i \) is clicked, \( #(j) \) is the number of times item \( j \) is clicked, and \( #(i, j) \) is the number of users who clicks both \( i \) and \( j \).

From Eq.(10) and Eq.(11) we can observe that, the item vectors and context vectors can be obtained by factorizing the matrix whose elements are defined in Eq.(11).

A practical issue arises here: for item pair \( (i, j) \) that are less often clicked by same user, \( PMI(i, j) \) is negative, or if they have never been clicked by the same user, \( #(i, j) = 0 \) and \( PMI(i, j) = -\infty \). However, a negative value of PMI does not necessarily imply that the items are not related. The reason may be because the number of items is very huge, and a user who click \( i \) may not know about the existence of \( j \). A common way in natural language processing is to replace the negative values by zeros to form the positive PMI (PPMI) matrix [2]. The PPMI matrix \( S \) whose elements are defined as follows.

\[ s_{ij} = \max\{PMI(i, j), 0\} \tag{12} \]

The item embedding model for implicit feedback can be summarized as follows: (1) construct an item-item matrix \( S \) regarding to items from the implicit feedback. We are ready to present idea to combine that model with the MF model for explicit feedback. Rather than performing two independent models: matrix factorization on user ratings, and item embedding on implicit feedback, we connect them in to a unified model which we describes below.

In item embedding above, item \( i \) will be represented by two vectors: item vector \( \rho_i \) and context vector \( \alpha_j \) which are derived from the observations of the PPMI matrix. Additionally, the model adds
an offset value \( e_i \) to the item vector \( \rho_j \) to capture the deviation from the item vector learned from the implicit feedback. The deviation \( e_i \) can be interpreted as the contribution of explicit feedback information into the representation of the items and is useful when the click information does not reflect the preference of a user to an item. For example, a user clicks an item many times and finally found that he/she does not like the item, giving a low rating scores.

The generative process for modeling the rating scores as well as the implicit feedback data is shown below.

(1) **Item embedding model**

(a) For each item \( i \): draw item vector \( \rho_i \) and context vector \( a_i \)

\[
\rho_i \sim N(0, \sigma_1^2 I)
\]

(13)

\[
a_i \sim N(0, \sigma_2^2 I)
\]

(14)

(b) For each pair \((i, j)\), draw \( s_{ij} \) of the PPMI matrix:

\[
s_{ij} \sim N(\rho_i^T a_j, \sigma_2^2)
\]

(15)

(2) **Rating model**

(a) For each user \( u \): draw user vector \( \theta_u \) and bias term \( b_u \)

\[
\theta_u \sim N(0, \sigma_2^2 I)
\]

(16)

\[
b_u \sim N(0, \sigma_3^2)
\]

(17)

(b) For each item \( i \): draw the deviation \( \epsilon_i \) and bias term \( c_i \)

\[
\epsilon_i \sim N(0, \sigma_2^2 I)
\]

(18)

\[
c_i \sim N(0, \sigma_2^2)
\]

(19)

then set \( \beta_i = \rho_i + \epsilon_i \)

(c) For each pair \((u, i)\): draw the rating score

\[
r_{ui} \sim N(\mu + b_u + c_i + \theta_u^T \beta_i, \sigma_2^2)
\]

(20)

### 3.3 Parameter learning

The parameters of the model \((\theta, \rho, \beta, \alpha, b, e)\) are learned by maximizing the log posterior distribution which is equivalent to minimizing the following error function.

\[
\mathcal{L}(\theta, \rho, \beta, \alpha, b, e) = \frac{1}{2} \sum_{(u, i) \in \mathcal{R}} \left[ r_{ui} - (\mu + b_u + c_i + \theta_u^T \beta_i) \right]^2 \\
+ \frac{\lambda}{2} \sum_{(i, j) \in \mathcal{S}} (s_{ij} - \rho_i^T a_j)^2 \\
+ \frac{\lambda_2}{2} \sum_{u=1}^N \|\theta_u\|_F^2 + \frac{\lambda_2}{2} \sum_{i=1}^M ||\beta_i - \rho_i||_F^2 \\
+ \frac{\lambda_3}{2} \sum_{i=1}^M ||\rho_i||_F^2 + \frac{\lambda_4}{2} \sum_{j=1}^N ||a_j||_F^2 \\
+ \frac{\lambda_5}{2} \sum_{u=1}^N \|\theta_u\|^2 + \frac{\lambda_6}{2} \sum_{i=1}^M ||a_i||^2
\]

(21)

where \( \mathcal{S} = \{(i, j)|s_{ij} > 0\}, \lambda = \sigma_1^2/\sigma_2^2, \lambda_2 = \sigma_2^2/\sigma_1^2, \lambda_3 = \sigma_2^2/\sigma_3^2, \lambda_4 = \sigma_2^2/\sigma_4^2, \lambda_5 = \sigma_3^2/\sigma_5^2, \lambda_6 = \sigma_2^2/\sigma_6^2 \), and \( \lambda = \sigma_2^2/\sigma_3^2 \)

Function in Eq.(21) is not convex with respect to \( \theta, \beta, \rho, \alpha, b, e \), but it is convex if we keep five of them fixed. Therefore, it can be solved using alternative least square (ALS) method, similar to the method described in [5].

For each user \( u \), at each iteration, we calculate the partial derivative of \( \mathcal{L}(\theta, \epsilon, \rho, \alpha) \) with respect to \( \theta_u \) while fixing other parameters. By setting this derivative to be zero:

\[
\frac{\partial \mathcal{L}}{\partial \theta_u} = 0
\]

we obtain the update rule of \( \theta_u \) as shown in Eq. (22).

\[
\theta_u = \left( \sum_{i \in \mathcal{R}_u} \beta_i^T \beta_i + \lambda I_d \right)^{-1} \left( \sum_{i \in \mathcal{R}_u} r_{ui} \beta_i \right)
\]

(22)

where \( \beta_i = \rho_i + \epsilon_i \) and \( I_d \) is the \( d \)-dimensional identity matrix.

By doing the same way, we can obtain the update rules for \( \epsilon_i, \rho_i \) and \( \alpha_j \) as shown in Eq.(23).

\[
\beta_i = \left( \sum_{u \in \mathcal{R}_i} \theta_u^T \rho_i + \lambda \beta_i I_d \right)^{-1} \left( \lambda \beta_i \theta_i + \lambda \sum_{j \in \mathcal{S}_i} s_{ij} \alpha_j \right)
\]

(23)

\[
\rho_i = \left( \lambda \sum_{j \in \mathcal{S}_i} \alpha_j a_j^T + \lambda I_d \right)^{-1} \left( \lambda \beta_i \rho_i + \lambda \sum_{j \in \mathcal{S}_i} s_{ij} \alpha_j \right)
\]

(24)

\[
\alpha_j = \left( \lambda \sum_{i \in \mathcal{S}_j} \beta_i \theta_i + \lambda I_d \right)^{-1} \left( \lambda \sum_{i \in \mathcal{S}_j} s_{ij} \beta_i \right)
\]

(25)

\[
b_u = \frac{\sum_{i \in \mathcal{R}_u} r_{ui} - (\mu |\mathcal{R}_u| + \sum_{i \in \mathcal{R}_u} c_i + \theta_u^T \beta_i)}{|\mathcal{R}_u| + \lambda b}
\]

(26)

\[
c_i = \frac{\sum_{u \in \mathcal{R}_u} r_{ui} - (\mu |\mathcal{R}_u| + \sum_{u \in \mathcal{R}_u} b_u + \theta_u^T \beta_i)}{|\mathcal{R}_u| + \lambda_c}
\]

(27)

where \( b_u = \mu + b_u + c_i, \mathcal{S}_i = \{ |s_{ij} > 0 \}, \mathcal{S}_j = \{ |s_{ij} > 0 \}, \mathcal{R}_u, \) again, is the set of items that \( u \) has given ratings, and \( \mathcal{R}_i \) is the set of users that \( u \) has given ratings to.

**Computational complexity.** For user vectors, as analyzed in [5], the complexity for updating \( N \) users in an iteration is \( O(d^2|\mathcal{R}| + d^3N) \), where \( |\mathcal{R}| \) is the number of non-zero entries of rating matrix \( R \). Since \( |\mathcal{R}| >> N, \) if \( d \) is small, this complexity is a linear in the size of the input matrix. For item vector updating, we also can easily show that the running time for updating \( M \) items in an iteration is \( O(d^2(|\mathcal{R}| + |\mathcal{S}|) + d^3M) \), where \( |\mathcal{S}| \) is the number of non-zero entries of matrix \( S \). We can see that the computational complexity linearly scales with the number of users and the number of items. Furthermore, this algorithm is easily to be parallelized to adapt with large scale data. For example, in updating user vectors \( \theta \), the update rule of user \( u \) is independent of other users’ vectors, therefore, we can compute \( \sum_{i} \beta_i \beta_i^T \) in advance, and update \( \theta_u \) in parallel.

### 3.4 Rating prediction

After learning parameters \( \theta, \rho, \beta, \alpha, b, e \), the proposed model can be used for predicting missing ratings. We consider two cases of rating predictions: **in-matrix prediction** and **out-matrix prediction**.

In-matrix prediction refers to the case that we predict the rating of user \( u \) to item \( i \), where \( i \) has not been rated by \( u \) but has been rated by at least one other users. Out-matrix refers to the case that we predict the rating of user \( u \) to item \( i \), where \( i \) has not been rated by any users (i.e., has implicit feedback only).

Let \( D \) be the observed data (observed rating scores), the unobserved \( r_{ui} \) can be estimated as in Eq.(28).
We split the rating data into two parts: 80% for training set and 20% for as ground-truth for testing. From the training set, we randomly pick 20% from the rating data and use as explicit feedback.

The statistical information about the datasets is given in Table 1.

Table 1: Statistical information of the datasets

|                | ML-1m  | ML-20m | Bookcrossing |
|----------------|--------|--------|--------------|
| # of users     | 6,040  | 138,493| 77,805       |
| # of items     | 3,706  | 26,744 | 185,973      |
| value of ratings| 1 - 5  | 0.5 - 5| 1 –10        |
| average rating  | 3.58   | 3.25   | 7.01         |
| # of ratings   | 1,000,209| 20,000,263| 357,246 |
| rating density (%) | 4.47   | 0.53   | 0.0029       |
| # of clicks    | -      | -      | 892,185      |
| click density (%) | -      | -      | 0.0062       |

Since Movielens datasets contain only explicit feedback, we artificially create the implicit feedback and explicit feedback data following [1]. For the implicit feedback, we use all the rating data by considering whether a user rated an item or not. In other words, the implicit feedback is obtained by binarizing the rating data. For explicit feedback, we randomly pick 20% from the rating data and use as explicit feedback.

4.2 Evaluation

We split the rating data into two parts: 80% for training set and 20% for as ground-truth for testing. From the training set, we randomly pick 10% as a validation set that will be used for model selection and checking stopping condition of the training phase. In evaluating the in-matrix prediction, we make sure that all the items in the test set appear in the training set (to ensure that all the items in the test set have at least one rating in the past). In evaluating out-matrix prediction, we make sure that none of the items in the test set appear in the training set (to ensure that none of the items in the test set have any rating in the past).

The model is trained on training dataset and the optimal parameters are obtained by using the validation set. The model with these optimal parameters are then used to predict ratings for user-item pairs that appear in the test set. We use Root Mean Square Error (RMSE), as the metric to measure the performance of the models. RMSE measures the deviation between the rating predicted by the model and the true ratings (given by the test set), and is defined as follows.

\[
RMSE = \sqrt{\frac{1}{|\text{Test}|} \sum_{(u,i) \in \text{Test}} (r_{ui} - \hat{r}_{ui})^2}
\]

where |Test| is the size of the test set. The smaller the value of RMSE on the test set is, the better the performance of the model is.

4.3 Competing methods

For in-matrix prediction. We compare our method with three factorization models as follows.

1. PMF [19]: a state-of-the-art method for rating prediction which we described in Section 2.
2. NMF (non-negative matrix factorization) [10]: a factorization model with the constraint that all the components of user factors and item factors must be non-negative.
3. SVD++ [6]: a factor model that exploits both explicit feedback and implicit feedback in rating prediction. The implicit feedback used in SVD++ is in the form: “who rated what?” which is inferred from the explicit feedback data.

We used the Librec 4, an open source library to run the above competing methods.

For out-matrix prediction. Because conventional collaborative filtering methods cannot predict ratings for such kind of items, we will compare our method with the method using user-basis average which estimates the rating of a user to an item by the average of the ratings that he gave to other items.

4.4 Parameter settings

In all settings, we set the dimension of the latent space to \(d = 20\). For PMF, NMF and SVD++, we used grid search to find the optimal values of the regularization terms that produce best performance on validation set. We found that, \(\lambda_u = \lambda Tenant 0.01\) give good performance. For our proposed method, we explored the parameters with different settings. First, we fixed \(\lambda = 1\) and used grid search for finding the optimal values of the remaining parameters that give good performance on validation set. We found that \(\lambda_\theta = \lambda_\phi = \lambda_\psi = \lambda_\alpha = \lambda_\beta = \lambda_\gamma = 10\) give good performance and used this setting in comparing with the competing methods. Second, we explored different values of \(\lambda\) while fixing the remaining parameters in order to study how the contribution of implicit
datasets. While SVD++ uses only "implicit feedback" data that is inferred from the explicit feedback, our method uses much more implicit feedback which comes from a different and independent source.

In Bookcrossing dataset, the test RMSE of NMF and PMF are far worse than SVD++ and our method. The reason is because the explicit feedback Bookcrossing data is extremely sparse (0.0029%) and explicit feedback only is not enough for prediction.

Table 3: Test RMSE of out-matrix prediction on three datasets.

| Methods      | ML-1m | ML-20m | Bookcrossing |
|--------------|-------|--------|--------------|
| User-basis average | 1.0528 | 0.9772 | 1.6553       |
| Our method   | 1.0044 | 0.9459 | 1.6428       |

The test RMSE of out-matrix prediction is given in Table 3. The results show that our method outperforms imputing ratings by user-basis average. From the result, we can see that the performance of our method in this case is worse than in in-matrix prediction. This is reasonable because this prediction is almost entirely based on the implicit feedback data.

4.5.1 Effect of the sparseness of explicit data. We study the performances of the methods on different level of sparsities of explicit feedback data. In the above experiment, from Movielens datasets, we create different subsets picking 10%, 50% and 90% from the original rating data and assume that only these amounts of ratings are available. We named these datasets: ML1-10, ML1-50, ML1-90 (obtained by picking 10%, 50%, 90% from ML-1m, respectively) and ML20-10, ML20-50, and ML20-90 (obtained by picking 10%, 50%, 90% from ML-20m, respectively). The densities of the rating matrices of these datasets are given in Table 4.

Table 4: Statistical information of some subsets drawn from Movielens data

| Original dataset | % selected | Density (%) |
|------------------|------------|-------------|
| ML1-10           | ML-1m      | 10%         | 0.3561      |
| ML1-50           | ML-1m      | 50%         | 1.6022      |
| ML1-90           | ML-1m      | 90%         | 2.8206      |
| ML20-10          | ML-20m     | 10%         | 0.1001      |
| ML20-50          | ML-20m     | 50%         | 0.2108      |
| ML20-90          | ML-20m     | 90%         | 0.3459      |

Table 5 and Table 6 show the test RMSE results of in-matrix and out-matrix prediction tasks on different subsets of the data. The experimental results show that our proposed method outperforms all competing methods in both in-matrix prediction and out-matrix prediction on two datasets over different levels sparsities of explicit feedback data. For all methods, the predicting accuracies increase with the density of explicit feedback. This is expected because the explicit feedback data is reliable for inferring users' preferences.

In all cases, the differences between our proposed method with the competing methods are most pronounced in the most sparse subsets (ML1-10 or ML20-10). This indicates the effectiveness of our proposed method for sparse data.

4.5.2 Impact of parameter $\lambda$. As in the Eq.21, parameter $\lambda$ controls the level of contribution of implicit feedback data to the model. If $\lambda = 0$, the model reduces to the original MF which uses explicit feedback only for modeling users and items. If $\lambda = \infty$, the model uses only information from the implicit feedback to model items. In this part, we vary $\lambda$ while fixing other parameters to study the effect of $\lambda$ on the accuracy of the model.

Table 7 shows the test RMSE of in-matrix prediction task of our proposed method when the $\lambda$ is varied. From the result, we can observe that the prediction performance is influenced significantly by the value of $\lambda$. For small values of $\lambda$, the test RMSE is relatively high, it decreases when $\lambda$ increases. However, when $\lambda$ goes over a certain threshold, the test RMSE starts increasing. This can be explained as follows. For a very small value of $\lambda$, the model mainly uses information from the explicit feedback which is too sparse to model the users and items. When the value of $\lambda$ becomes very large, the model mainly use the implicit feedback data for modeling the items therefore is not reliable. The best values of $\lambda$ should balance the contribution of implicit and explicit feedback.

4.5.3 Impact of parameter $\lambda_{\beta}$. $\lambda_{\beta}$ is the parameter that controls the deviation of item latent vector $\beta_i$ from the item embedding vector ($\rho_i$). When $\lambda_{\beta}$ is small, the value of $\beta_i$ is allowed to diverge from $\rho_i$; in this case, the information for modeling item $i$ mainly comes from explicit feedback. On the other hand, when $\lambda_{\beta}$ increases, $\rho_i$ becomes closer to $\rho_i$; in this case the item vectors mainly come from the embedding model of implicit feedback.

Table 8 shows the test RMSE of in-matrix prediction when $\lambda_{\beta}$ is varied while other parameters are fixed. From the result we can...
Table 5: Test RMSE of in-matrix prediction for different subsets of ML-1m and ML-20m dataset

| Methods | ML-1m  |   |   | ML-20m  |   |   |
|---------|--------|---|---|---------|---|---|
|         | ML1-10 | ML1-50 | ML1-90 |         | ML20-10 | ML20-50 | ML20-90 |
| PMF     | 1.1026 | 0.9424 | 0.8983 |         | 1.0071 | 0.8663 | 0.8441 |
| NMF     | 1.0807 | 0.9566 | 0.9451 |         | 1.0838 | 0.9101 | 0.8935 |
| SVD++   | 0.9825 | 0.9066 | 0.8871 |         | 0.8947 | 0.8348 | 0.8191 |
| Our method | **0.9371** | **0.8719** | **0.8498** |         | **0.8767** | **0.8299** | **0.8024** |

Table 6: Test RMSE of out-matrix prediction for different subsets of ML-1m and ML-20m dataset

| Methods | ML-1m  |   |   | ML-20m  |   |   |
|---------|--------|---|---|---------|---|---|
|         | ML1-10 | ML1-50 | ML1-90 |         | ML20-10 | ML20-50 | ML20-90 |
| User average | 1.0946 | 1.0335 | 1.039 |         | 1.0088 | 0.9562 | 0.9714 |
| Our method | **1.0312** | **1.0059** | **1.0132** |         | **0.9729** | **0.9422** | **0.9714** |

Table 7: Test RMSE of in-matrix prediction task on different subsets of ML-1m dataset corresponding to different values of $\lambda$

| $\lambda$ | 0.01 | 1 | 10 | 30 | 50 | 100 | 200 | 1000 |
|-----------|------|---|---|---|----|-----|-----|------|
| ML1-10    | 0.9614 | 0.9366 | 0.9340 | 0.9326 | 0.9316 | 0.9318 | 0.9336 | 0.9348 |
| ML1-50    | 0.9339 | 0.9024 | 0.8887 | 0.8885 | 0.8882 | 0.8883 | 0.8887 | 0.8892 |
| ML1-90    | 0.8869 | 0.8689 | 0.8634 | 0.8637 | 0.8636 | 0.8637 | 0.8639 | 0.8645 |

Table 8: Test RMSE of in-matrix prediction task on different subsets of ML-1m dataset corresponding to different values of $\lambda$ and $\beta$

| $\lambda$ | 0.1 | 1.0 | 10.0 | 20.0 | 50.0 | 100.0 | 1000.0 |
|-----------|-----|-----|------|------|------|-------|--------|
| ML1-10    | 1.1301 | 0.9971 | 0.9318 | 0.9381 | 0.9527 | 0.9651 | 0.9924 |
| ML1-50    | 1.1107 | 0.9963 | 0.8911 | 0.8756 | 0.8723 | 0.8769 | 0.8885 |
| ML1-90    | 0.9798 | 0.9193 | 0.8634 | 0.8545 | 0.8512 | 0.8539 | 0.8626 |

Observe that, for small values of $\lambda$ and $\beta$, the model produces low prediction accuracy (high test RMSE). The reason is when $\lambda$ is small, the model mostly relies on the explicit feedback which is very sparse and can not model users and items well. When $\lambda$ increases, the model starts using implicit feedback for prediction, the accuracy will increase. However, when $\lambda$ reaches a certain threshold, the accuracy starts decreasing. This is because when $\lambda$ is too large, the representations of items mainly come from implicit feedback, and therefore the model becomes less reliable to model the rating data.

5 RELATED WORK

The idea of using implicit feedback to boost the performance of rating prediction is first introduced in [1] and SVD++ [6]. In these models, the authors integrate implicit feedback which is in the form "who rates what" into the factorization model. The implicit feedback that these models used is inferred from the rating data, therefore they can not model an item if it does not have any rating. Our approach is different, we use implicit feedback as an independent data source, therefore our model can model an item even if it does not have any rating. Another difference is that, the implicit feedback used in [1] and [6] is inferred from rating data, therefore it is also very sparse. The implicit feedback used in our model is more dense and we can exploit more information from the implicit feedback.

Co-ranking [13] combines explicit and implicit feedback by treating explicit feedback as a special kind of implicit feedback. The explicit feedback is normalized into the range $[0, 1]$ and is summed up to implicit feedback matrix with a fixed proportion to form a single matrix. This matrix is then factorized to obtain the latent vectors of users and items. The main difference of Co-ranking and our method is that, the contribution of implicit feedback and explicit feedback in the representation of items is equal for every item, while in our approach this proportion is dynamic. For an item to which there are a lot of ratings available, its representation mainly comes from the explicit feedback. On the other hand, if an item has few or does not have any rating data, its representation mainly comes from the implicit feedback data. On the other hand, if an item has few or does not have any rating data, its representation mainly comes from the implicit feedback data.

Wang et. al. [22] proposed Expectation-Maximization Collaborative Filtering (EMCF) which exploits both implicit and explicit feedback for recommendation. For predicting ratings for an item to which rating data is not available, the rating is inferred from ratings of its neighbor regarding to click data. However, the algorithm is based on an iterative Expectation-Maximization (EM) in which E-phase is a matrix factorization model. In other words, it needs multiple times of matrix factorization and therefore is not efficient in computation.

Collective matrix factorization (CMF) [21] proposed a framework for factorizing multiple related matrices simultaneously, in order to exploit information from multiple sources. For example,
if the item-genre matrix exists, one can factorize both user-item and item-genre matrices in a shared latent space. This approach is able to incorporate the side information (e.g., genre information of items) into the latent factor model. Our model is a special case of CMF with the rating matrix and item-item co-occurrence matrix.

6 DISCUSSION AND FUTURE WORK

In this paper, we proposed a probabilistic model that combines explicit feedback and implicit feedback for rating prediction task. The model is a combination of two models: MF for explicit feedback and item embedding for implicit feedback. The experimental results showed that our proposed method improved the accuracy of rating prediction for three real world datasets. Our method also can efficiently learn the latent representations items two which the rating data is not available and efficiently predict the missing ratings for them.

We plan to explore several ways of extending or improving this work. In this model, the optimal hyper-parameters are found by performing grid search. It is very time consuming for large datasets or if we increase the range of choices for hyper-parameters. One direction to improve this work is to develop a full Bayesian model that treats hyper-parameters as random variables and we can perform inferring posterior distribution over hyper-parameters.

The second direction we are planning to pursue is to develop an online learning algorithm, which updates user and item vectors when new data are collected without retraining the model from the beginning.

REFERENCES

[1] Robert M. Bell and Yehuda Koren. 2007. Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. In ICDM. IEEE Computer Society, 43–52.
[2] John Bullinaria and Joseph Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. Behavior Research Methods 39, 3 (2007), 510–526.
[3] K. W. Church and P. Hanks. 1990. Word Association Norms, Mutual Information, and Lexicography. Computational Linguistics 1, 16 (1990), 22–29.
[4] Prem Gopalan, Laurent Charlin, and David M. Blei. 2014. Content-based recommendations with Poisson factorization. In NIPS. 3176–3184.
[5] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on, IEEE, 263–272.
[6] Y. Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 426–434.
[7] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. Commun. ACM 53, 4 (2010), 89–97.
[8] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. Commun. ACM 53, 4 (2010), 89–97.
[9] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In ICML, Vol. 14. 1188–1196.
[10] Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for Non-negative Matrix Factorization. In NIPS. 556–562.
[11] Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. In Advances in Neural Information Processing Systems 27. Curran Associates, Inc., 2177–2185.
[12] Shaohua Li, Jun Zhu, and Chunyan Miao. 2015. A Generative Word Embedding Model and its Low Rank Positive Semidefinite Solution. In EMNLP. The Association for Computational Linguistics, 1599–1609.
[13] Nathan Nan Liu, Evan Wei Xiang, Min Zhao, and Qiang Yang 0001. 2010. Unifying explicit and implicit feedback for collaborative filtering. In CIKM. ACM, 1445–1448.
[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013).
[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality.

In NIPS. 3111–3119.
[16] Xia Ning, Christian Dersiories, and George Karypis. 2015. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In Recommender Systems Handbook. Springer, 37–76.
[17] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. 2008. One-Class Collaborative Filtering. In IEEE International Conference on Data Mining (ICDM 2008) 502–511.
[18] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence.
[19] Ruslan Salakhutdinov and Andryr Mnih. 2008. Probabilistic Matrix Factorization. In Advances in Neural Information Processing Systems, Vol. 20.
[20] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl. 2001. Item-based collaborative filtering recommendation algorithms. In World Wide Web 285–295.
[21] Ajit Paul Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In KDD. ACM, 650–658.
[22] Bin Wang, Mohammadreza Rahimi, Dequan Zhou, and Xin Wang. 2012. Expectation-Maximization Collaborative Filtering with Explicit and Implicit Feedback. In PAKDD (1) (Lecture Notes in Computer Science), Vol. 7301. Springer, 604–616.
[23] Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In KDD. ACM, 448–456.
[24] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In KDD. ACM, 1235–1244.