Knowledge-Based Systems 97 (2016) 188–202

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model

Antonio Hernando[a,*], Jesús Bobadilla[a], Fernando Ortega[a]

Universidad Politécnica de Madrid, Carretera de Valencia km 7, 28031 Madrid, Spain

ARTICLE INFO

Article history:
Received 12 April 2015
Revised 16 November 2015
Accepted 25 December 2015
Available online 6 January 2016

Keywords:
Recommender systems
Collaborative filtering
Matrix factorization
Graphical probabilistic models

ABSTRACT

In this paper we present a novel technique for predicting the tastes of users in recommender systems based on collaborative filtering. Our technique is based on factorizing the rating matrix into two non negative matrices whose components lie within the range [0, 1] with an understandable probabilistic meaning. Thanks to this decomposition we can accurately predict the ratings of users, find out some groups of users with the same tastes, as well as justify and understand the recommendations our technique provides.

1. Introduction

Recommender systems are systems able to provide personalized recommendations to users [22]. Recommender Systems have been used in different domains, such as music [23], television [24], books [25], e-learning [26] or e-commerce [27]. However, most research papers have focused on movie recommendations.

Recommender systems are usually classified into two main categories according to the input of these systems:

- Based on contents. Recommender systems of this kind require that the items are described by means of some features or a verbal description [2,4]. Besides, such recommender systems need that users inform about their preferences of the kind of items they like. This can be implicitly obtained by observing the kind of items users consume.
- Based on collaborative filtering. Such recommender systems use a rating matrix $\mathcal{M}$ in which each user u provides information about how much he likes some items (see Tables 1 and 9 for examples of rating matrices). The recommender system does not use information about features of users or of the items. In this way, recommender systems of this kind detect the taste of users through the ratings users have already made. In this paper we will focus on Recommender Systems based on collaborative filtering.

Recommender systems based on collaborative filtering can be classified on behalf of the kind of algorithm they use to predict the tastes of users.

- Based on Memory. Memory-based recommender systems basically use the K-Nearest Neighbor algorithm (K-NN algorithm) for predicting the tastes of users. These recommender systems involve very interesting features:
  - The algorithm is very intuitive, since it is based on the following idea: in order to recommend items to user u, the algorithm finds out the users with tastes similar to those of u, called the neighbors of u, and recommends the user u those items that some of these neighbors of u have already rated highly. Because of the intuitiveness of the algorithm, such recommender system are able to explain the recommendations they provide [17]. Indeed, the following property holds in the K-NN algorithm:

Proposition 1. *If the recommender system predicts that a user u will like the item i, then there are some users with the same taste as u who have very positively rated the item i.*

  - The quality of the predictions and recommendations can be quite good, provided that a suitable similarity function is used to measure how similar two users are on behalf of their tastes. Besides, we can measure the reliability of the predictions and recommendations [16].

Nevertheless, a major drawback of memory-based recommender systems is that the they do not use a scalable algorithm for predicting and recommending items. Therefore, these recommender systems are not suitable when dealing with a great

* Corresponding author. Tel.: +34 658537247.
E-mail addresses: antonio.hernando@upm.es, Ahernando@etsisi.upm.es (A. Hernando), jesus.bobadilla@upm.es (J. Bobadilla).

http://dx.doi.org/10.1016/j.knosys.2015.12.018
0950-7051/

Table 1
First example of rating matrix.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
U_1	5	5	5	5	5	•	•	•	•	•	•	•	•	•	•
U_2	5	5	5	5	5	•	•	•	•	•	•	•	•	•	•
U_3	5	5	•	5	5	•	•	•	•	•	•	•	•	•	•
U_4	•	•	•	•	•	5	5	5	5	5	•	•	•	•	•
U_5	•	•	•	•	•	5	5	5	5	5	•	•	•	•	•
U_6	•	•	•	•	•	5	5	5	5	5	•	•	•	•	•
U_7	•	•	•	•	•	•	•	•	•	•	5	5	5	5	5
U_8	•	•	•	•	•	•	•	•	•	•	5	5	5	5	5
U_9	•	•	•	•	•	•	•	•	•	•	5	5	5	5	5
U_{10}	5	5	5	5	5	•	•	•	•	•	•	•	•	•	•
U_{11}	•	•	•	•	•	5	5	5	5	5	•	•	•	•	•

number of items and users. In order to avoid this important drawback, we will here focus on model-based recommender systems.

- Based on models. Model-based recommender systems are based on the idea of using a model to predict the ratings that users make. Since such recommender systems involve scalable algorithms for recommending, they are very advisable when using large recommender systems with a great number of users or items. These recommender systems require a learning phase for finding out the user and item matrices before the recommender system starts. However, a model-based recommender system can predict the ratings of users very quickly, once the learning phase is finished.
  Among all models proposed, those providing better accuracy in the predictions are the one based on factoring the rating matrix into two matrices [6]: one related to items and another one related to users. This factorization technique is based on the idea of finding out unknown K latent factors that allow to predict the ratings of users by solving an optimization problem (the learning phase). Once the optimization problem is solved (the learning phase is through), each user u and each item i are respectively associated to vectors of K components $\vec{a}_u$ and $\vec{b}_i$ that are used to predict, $p_{u,i}$, the rating that user u would make on item i:

$$p_{u,i} = \vec{a}_u \cdot \vec{b}_i$$

  Besides, this factorization technique provides significant improvement of the quality of predictions and recommendations over the ones based on memory.
  The paper [5] gives an interpretation of the optimization problem posed during the learning phase in probabilistic terms: they consider that vectors $\vec{a}_u$, $\vec{b}_i$ and the ratings $r_{u,i}$ follow a Gaussian distribution. However, the components of vectors $\vec{a}_u$, $\vec{b}_i$ are still very hard to understand: since their components can take arbitrary (even negative) values, they do not have any straightforward probabilistic interpretation. More sophisticated versions of this model have been proposed considering bias of user and items, and making use of information about the users' preferences [6]. However, all these models also suffer from an important drawback: they cannot justify the predictions the models make since the components of vectors $\vec{a}_u$, $\vec{b}_i$ are hard to understand. Besides, this technique does not fulfill Proposition 1 (we will see some examples of this in Section 3). Recently, [19] deals with a new probabilistic model based on the Poisson distribution, of interest for recommender systems with other kinds of input: in these, instead of considering that the input is a rating matrix where each user explicitly evaluates some items, a matrix is considered indicating how many times each user has consumed each item (without indicating whether users liked the item or not). In this model, vectors $\vec{a}_u$ and $\vec{b}_i$ are more easily understandable, since they only take positive values. However, since this model is not suitably designed for a rating matrix as input (here the input is a matrix of consumed items), the resulting recommender systems do not provide good predictions about the ratings of users (according to MAE).

In our paper we will present a novel technique for factoring the rating matrix, preserving the advantages of the classical factorization technique:

- Just like in classical matrix factorization, we also consider the existence of K latent factors explaining the ratings that users make. In this way, we associate a K dimensional vector $\vec{a}_u = (a_{u,1}, \ldots, a_{u,K})$ for each user u and a K dimensional vector $\vec{b}_i = (b_{1,i}, \ldots, b_{K,i})$ for each item i. However, as we will see next, the components of these vectors in our model present significant advantages over the ones in the classical matrix factorization:
- Since we present a model for making predictions, our technique is completely scalable.
- Our model provides good quality of predictions and recommendations. Indeed, as we will see, according to our results, the quality of recommendations can be considered still better as those made with the technique [5].

In addition to keeping the advantages of the classical matrix factorization, our technique provides additional advantages:

- While $a_{u,k}$ and $b_{k,i}$ may take arbitrary real values in the classical matrix factorization, in our model $a_{u,k}$ and $b_{k,i}$ are bound to lie within the range [0, 1] .
- Unlike classical matrix factorization, here the values $a_{u,k}$ and $b_{k,i}$ have a understandable probabilistic interpretation:
  - Latent factors represent groups of users who share the same tastes in our system. In this way, the parameter K indicates the number of such groups in our database.
  - The value $a_{u,k}$ represents the probability that user u belongs to the group k of users. In this way, the following holds:

$$\sum_{i=1}^{K} a_{u,k} = 1$$

  - The value $b_{i,k}$ represents the probability that users in the group k like item i.
- In order to get accurate predictions in the recommender system, the parameter K is lower in our model than in the classical matrix factorization. Consequently, vectors $\vec{a}_u$ and $\vec{b}_i$ have a lower dimension in our model, involving more efficiency in memory as well as more efficiency when performing operations with these vectors (when performing the learning phase or when predicting if a user will like the item $p_{u,i} = \vec{a}_u \cdot \vec{b}_i$).

- Unlike classical matrix factorization, here the vector $\vec{a}_u$ is very sparse: that is to say, very few components of the vector $\vec{a}_u$ take high values, while a lot of components take values very close to 0 (which can be regarded as 0 for practical reasons). Consequently, the prediction of the ratings $p_{u,i} = \vec{a}_u \cdot \vec{b}_i = \sum_{k=1}^{K} a_{u,k} \cdot b_{k,i}$ is usually done very quickly, since very few components of this summation are involved in the prediction.
- According to the previously discussed properties of $\vec{a}_u$ and $\vec{b}_i$ in our model, we will see that our technique fulfills Proposition 1: if the recommender system predicts that a user u will like the item i, then there are some users with the same taste as u who have very positively rated the item i. This property of our model (which does not hold in classical factorization) allows us to justify and understand all the recommendations our technique provides.

The paper is structured in the following way. In Section 2 we discuss techniques related to ours in recommender systems. In Section 3 we present some clarifying examples showing the main drawbacks of the techniques based on the matrix factorization taken as a reference in collaborative filtering [5,6]. In Section 4 we present the probabilistic model proposed in this paper and the learning algorithm we use to infer the matrices. In Section 5 we interpret the meaning of the output of the algorithm and we see how this allows us to justify the recommendations in the same way as K-NN does. In Section 6 we evaluate the accuracy of the predictions in our technique. Finally, in Section 7 we set our conclusions.

2. Related work

In this paper we present a technique which factorizes the rating matrix into two matrices: a matrix associated to users (each row represents the user vector associated to each user); another one associated to items (each column represents the item vector associated to each item).

The idea of factoring the matrix is well-known in the field of Machine Learning through techniques like Principal Component Analysis or Non Negative Matrix Factoring. These techniques have also been applied to the subfield of recommender systems.

In content-based recommender systems these techniques have also been applied in a matrix representing verbal descriptions of items: in this matrix, each component ($x_{i,j}$) describes how many times each word w_i appears in the description of each item j. Unlike the rating matrix used in collaborative filtering, all the terms of this matrix are already known (although it contains a lot of 0 values). Next, we enumerate the main techniques for decomposing this matrix:

- Latent Semantic Analysis (LSA) [4]. This technique is based on decomposing the description matrix into two matrices taking into account latent factors explaining the frequency of each word in the description of each item. One of these matrices is related to words (each row is the word vector a_w of each word) and the other one is related to the description of items (each column b_i is the item vector of each word). Like in [5,6], since the components of these matrices may take arbitrary real values, they may result to be very hard to understand.
- Probabilistic Latent Semantic Analysis (pLSA) [3]. This technique is completely related to the non-negative factorization of the description matrix of items [8]. Unlike latent semantic analysis, the components of the matrices are constrained to take positive values. By means of this technique, we model the probability that a word may appear in the description of a document. Due to the constraint that components of the matrices are non-negative, the meaning of these matrices can be understood and explained. Indeed, latent factors can be regarded as clusters of items and the components of each matrix can be regarded as: the probability that a word appears in the description of each cluster of items; and the probability that each item belongs to each cluster of items.
- Latent Dirichlet Allocation (LDA) [2]. This technique can be regarded as an extension of pLSA through a Bayesian probabilistic model that avoids the overfitting that pLSA suffers. This technique is based on a Bayesian graphic probabilistic model. Like pLSA, this technique calculates the probability that a word appears in a document. This technique outputs two separate matrices: one related to each item, informing about the probability that an item belongs to a cluster of items; and another one related to words, informing about the probability that each word appears in the description of each cluster of items.

All of these techniques, while originally intended for content-based recommender systems, have also been applied to collaborative filtering recommender systems. However, regarding the latter, these techniques have not been successfully applied in terms of accuracy of the predictions of the ratings (MAE) because of the nature of the rating matrix: whereas there is a lot of unknown components in the rating matrix within collaborative filtering recommender systems (representing that we do not know whether the user likes the item or not), all the components of the matrix used in content-based recommender systems are thoroughly known. In this way, applying the previous techniques (like LSA, pLSA or LDA) to the rating matrix implicitly involves filling the matrix with 0s for the unknown values: in other terms, stating that the user actually *dislikes* these items, when we do not known indeed whether the user likes each of the items or not. However, although these techniques may not be suitable for predicting the ratings, they are much more successful if using other kind of matrices representing the number of times that a user consumes an item (which are very interesting for some domains, like music). In this case, these techniques (like Poisson Matrix Factoring [19]) very successfully predict how many times a user will consume an item.

Classical Matrix factorization techniques [5,6] are successful regarding collaborative filtering because they do consider that the lack of rating is not equivalent to a zero value. They pose an optimization problem that only takes into account the known ratings. However, these techniques present the same problems as LSA in content-based recommender systems: they do not constrain the components of the decomposition matrices to be non negative involving the difficulty of understanding the meaning of each component and the predictions.

Non-negative matrix factorization has been subject to much study in different fields [28–30,32]. There have also been made attempts to apply non-negative matrix factorization [8] to collaborative filtering, which have been shown to be equivalent to applying pLSA [20]. However, these attempts are not completely successful in predicting the ratings of users because, as stated above, they implicitly consider that the lack of rating is equivalent to the fact that a user completely dislikes an item. The model pLSA [3] has been extended for collaborative filtering using EM algorithm and considering that the ratings follow either a normal distribution or a multinomial distribution [18]. However, their results are far from being competitive with those in [5,6]: unlike our technique, the models proposed do not use a Bayesian framework and tend to have overfitting (in the same way as pLSA [3]). In this way, classical matrix factorization techniques [5,6] are the reference for matrix factorization techniques used for collaborative filtering recommender systems. Indeed, many extensions have been proposed for considering other kinds of information besides the rating matrix, like implicit information on the preference of users [15] or content-based information [13], or social information [14]. Although attempts have been made to use non-negative matrix

Table 2
Predictions of the ratings according to the classical matrix factorization.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
U_1	**5.01**	**5.01**	**5.01**	**5.01**	**5.01**	5.01	5.01	5.01	5.01	5.01	5.01	5.01	5.01	5.01	5.01
U_2	**5.01**	**5.01**	**5.01**	**5.01**	**5.01**	5.01	5.01	5.01	5.01	5.01	5.01	5.01	5.01	5.01	5.01
U_3	**5.00**	**5.00**	5.00	**5.00**	**5.00**	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
U_4	5.23	5.23	5.23	5.23	5.23	**5.23**	**5.23**	**5.23**	**5.23**	**5.23**	5.23	5.23	5.23	5.23	5.23
U_5	4.96	4.96	4.96	4.96	4.96	**4.96**	**4.96**	**4.96**	**4.96**	**4.96**	4.96	4.96	4.96	4.96	4.96
U_6	4.96	4.96	4.96	4.96	4.96	**4.96**	**4.96**	**4.96**	**4.96**	**4.96**	4.96	4.96	4.96	4.96	4.96
U_7	4.88	4.88	4.88	4.88	4.88	4.88	4.88	4.88	4.88	4.88	**4.88**	**4.88**	**4.88**	**4.88**	**4.88**
U_8	4.85	4.85	4.85	4.85	4.85	4.85	4.85	4.85	4.85	4.85	**4.85**	**4.85**	**4.85**	**4.85**	**4.85**
U_9	4.88	4.88	4.88	4.88	4.88	4.88	4.88	4.88	4.88	4.88	**4.88**	**4.88**	**4.88**	**4.88**	**4.88**
U_{10}	**5.01**	**5.01**	**5.01**	**5.01**	**5.01**	5.01	5.01	5.01	5.01	5.01	5.01	5.01	5.01	5.01	5.01
U_{11}	4.95	4.95	4.95	4.95	4.95	**4.95**	**4.95**	**4.95**	**4.95**	**4.95**	4.95	4.95	4.95	4.95	4.95

factorization [31] in recent years, unlike our approach, they do not use a complete variational Bayesian approach for this problem.

This paper can be regarded as a successful alternative to [8] for decomposing the rating matrix in non-negative matrices. Indeed, the components of these matrices lie within [0, 1], thus providing an understandable probabilistic meaning. Unlike the previous attempts of non-negative factorization, our model can very accurately predict the users' ratings. In order to avoid overfitting, our technique, unlike [18], is based on a completely Bayesian probabilistic framework.

3. Motivation

In this section, we will show some examples of rating matrices illustrating the disadvantages of the classical matrix factorization.

3.1. Example 1

In Table 1 we show an example of matrix of ratings with $N = 11$ users and $M = 15$ items. As may be appreciated in this rating matrix, there are clearly three groups of users sharing the same tastes: $\{U_1, U_2, U_3, U_{10}\}$, $\{U_4, U_5, U_6, U_{11}\}$ and $\{U_7, U_8, U_9\}$.

According to this rating matrix, we can intuitively predict that user U_3 will like the item I_3. However, it seems reasonable that there is no possible way to accurately predict any other unknown ratings in this example. Indeed, if the recommender system were asked, the recommender should reasonably state that the user U_1 is not specially interested in the item I_8. Consequently, if it is forced to predict, the recommender system should predict, in a conservative way, that the rating would be 3 (over a scale from 1 to 5) in order to minimize the average error.

Classic matrix factorization [5,6] is a technique which provides very accurate predictions. When applying it to this example, we obtain the predictions described in Table 2. As we expected, this technique has predicted that the user U_3 will like the item I_3 (with rating 5). However, it also predicts that user U_3 will like (with high value) the rest of unrated items, which, as stated above, seems not to be reasonable and is very hard to justify. Indeed, this model does not fulfill Proposition 1 in many predictions: for example, this technique predicts that user U_1 will like the item I_6 and, however, there are no users with the same taste as U_1 who have positively rated the item I_6.

In Tables 3 and4 we describe respectively the user and item matrices that the classic matrix factorization outputs. As may be seen, these matrices are not intuitive to interpret and it is not possible to infer the three groups of users from these matrices.

Using the standard Non-Negative Matrix factorization [8] with the rating matrix in Table 1, we would have obtained the matrix

Table 3
Matrix associated to users according to the classic matrix factorization.

	F_1	F_2	F3
U_1	0.005955	−0.028157	−0.010542
U_2	0.000836	−0.025898	−0.004726
U_3	−0.028857	−0.052094	0.025865
U_4	−1.026521	0.209566	0.185136
U_5	0.173929	−0.036068	−0.121470
U_6	0.178398	−0.033956	−0.081207
U_7	0.454605	−0.058759	−0.026684
U_8	−0.576575	−0.010821	0.055920
U_9	0.454591	−0.056812	−0.030514
U_{10}	0.013350	−0.029253	−0.017673
U_{11}	0.193889	−0.038002	0.005824

in Table 5. As may be seen, although the predictions of this technique are very accurate for the known ratings, the predictions of the unknown ratings are very low. This problem lies in the fact that this technique implicitly consider that the lack of rating on an item is equivalent to disliking it.

When applying our technique in Table 1 we obtain the predictions described in Table 6. Like the classic matrix factorization, our technique also predicts that the user U_3 will like the item I_3 (with rating 5). Besides, as we expected, our technique does not predict that user U_3 will like the rest of unrated items: the predictions for these ratings is 3 in order to minimize the average error.

Like the classic matrix factorization, our technique also provides both user and item matrices. However, unlike the classic matrix factorization, here these matrices are very easy to understand and are completely related to the groups of users:

- In Table 7 we show the user matrix obtained with our technique. Each value $a_{u,k}$ of this matrix indicates the probability that the user u belongs to the group k. According to this matrix, we can easily derive each group of users: group 1 with $\{U_1, U_2, U_3, U_{10}\}$; ; group 2 with $\{U_4, U_5, U_6\}$; and group 3 with $\{U_7, U_8, U_9\}$.
- In Table 8 we show the item matrix obtained with our technique. Each term $b_{i,k}$ of this matrix indicates the probability that users in group k like the item i. According to this matrix, we can easily derive the following: users in group 1 like the items $\{I_1, I_2, I_3, I_4, I_5\}$; users in group 2 like the items $\{I_6, I_7, I_8, I_9, I_{10}\}$; and users in group 3 like the items $\{I_{11}, I_{12}, I_{13}, I_{14}, I_{15}\}$.

3.2. Example 2

In Table 9 we show another example of rating matrix with the same dimension: $N = 11$ users and $M = 15$ items. We can recognize

Table 4
Matrix associated to items according to the classic matrix factorization.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
F_1	0.05	−0.09	−0.06	0.05	−0.06	0.20	−1.12	0.19	0.17	0.17	−0.01	−0.83	−0.03	0.76	−0.05
F_2	0.00	0.06	0.05	−0.07	−0.03	0.04	0.21	−0.06	−0.10	−0.12	0.02	0.14	0.14	0.03	−0.14
F_3	0.08	−0.01	−0.10	0.09	−0.01	0.01	0.27	0.07	0.01	0.01	0.10	−0.01	−0.07	−0.09	−0.15

Table 5
Predictions of the ratings according to the non-negative matrix factorization [8].

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
U_1	**5.26**	**5.26**	**5.26**	**5.26**	**5.26**	0	0	0	0	0	0	0	0	0	0
U_2	**5.26**	**5.26**	**5.26**	**5.26**	**5.26**	0	0	0	0	0	0	0	0	0	0
U_3	**4.21**	**4.21**	3.15	**4.21**	**4.21**	0	0	0	0	0	0	0	0	0	0
U_4	0	0	0	0	0	**5.26**	**5.26**	**5.26**	**5.26**	**5.26**	0	0	0	0	0
U_5	0	0	0	0	0	**5.26**	**5.26**	**5.26**	**5.26**	**5.26**	0	0	0	0	0
U_6	0	0	0	0	0	**5.26**	**5.26**	**5.26**	**5.26**	**5.26**	0	0	0	0	0
U_7	0	0	0	0	0	0	0	0	0	0	**5.26**	**5.26**	**5.26**	**5.26**	**5.26**
U_8	0	0	0	0	0	0	0	0	0	0	**5.26**	**5.26**	**5.26**	**5.26**	**5.26**
U_9	0	0	0	0	0	0	0	0	0	0	**5.26**	**5.26**	**5.26**	**5.26**	**5.26**
U_{10}	**5.26**	**5.26**	**5.26**	**5.26**	**5.26**	0	0	0	0	0	0	0	0	0	0
U_{11}	0	0	0	0	0	**5.26**	**5.26**	**5.26**	**5.26**	**5.26**	0	0	0	0	0

Table 6
Predictions of the ratings according to our technique.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
U_1	**5**	**5**	**5**	**5**	**5**	3	3	3	3	3	3	3	3	3	3
U_2	**5**	**5**	**5**	**5**	**5**	3	3	3	3	3	3	3	3	3	3
U_3	**5**	**5**	5	**5**	**5**	3	3	3	3	3	3	3	3	3	3
U_4	3	3	3	3	3	**5**	**5**	**5**	**5**	**5**	3	3	3	3	3
U_5	3	3	3	3	3	**5**	**5**	**5**	**5**	**5**	3	3	3	3	3
U_6	3	3	3	3	3	**5**	**5**	**5**	**5**	**5**	3	3	3	3	3
U_7	3	3	3	3	3	3	3	3	3	3	**5**	**5**	**5**	**5**	**5**
U_8	3	3	3	3	3	3	3	3	3	3	**5**	**5**	**5**	**5**	**5**
U_9	3	3	3	3	3	3	3	3	3	3	**5**	**5**	**5**	**5**	**5**
U_{10}	**5**	**5**	**5**	**5**	**5**	3	3	3	3	3	3	3	3	3	3
U_{11}	3	3	3	3	3	**5**	**5**	**5**	**5**	**5**	3	3	3	3	3

Table 7
Matrix associated to users according to our technique. Each term of this matrix indicates the probability that a user lies in a group. As may be seen, there are three groups: group 1 with $\{U_1, U_2, U_3, U_{10}\}$; group 2 with $\{U_4, U_5, U_6\}$; and group 3 with $\{U_7, U_8, U_9\}$.

	F_1	F_2	F_3
U_1	**0.96**	0.02	0.02
U_2	**0.96**	0.02	0.02
U_3	**0.96**	0.02	0.02
U_4	0.02	**0.96**	0.02
U_5	0.02	**0.96**	0.02
U_6	0.02	**0.96**	0.02
U_7	0.02	0.02	**0.96**
U_8	0.02	0.02	**0.96**
U_9	0.02	0.02	**0.96**
U_{10}	**0.96**	0.02	0.02
U_{11}	0.02	**0.96**	0.02

in this rating matrix that there are three groups of users again: $\{U_1, U_2, U_3\}$, $\{U_4, U_5, U_6, U_{11}\}$ and $\{U_7, U_8, U_9\}$. There are some doubts about user U_{10}, since it seems to belong both to group of $\{U_1, U_2, U_3\}$ (with higher probability) and to group $\{U_4, U_5, U_6, U_{11}\}$ (with less probability).

According to this rating matrix, we can intuitively predict that the user U_3 will like the item I_3 and the user U_5 will like the item I_8. Besides, it seems reasonable to think that we cannot accurately predict some unknown ratings: for example, the rating of the user U_6 on the item I_{15}, or the rating of the user U_{11} on the item I_{11}. Consequently, if it is forced to predict, the recommender system should predict, in a conservative way, that the rating would be 3 (over a scale from 1 to 5) in order to minimize the average error.

When applying classic matrix factorization to this rating matrix we obtain the predictions shown in Table 10 (with $K = 3$). As we expected, this technique has predicted that the user U_3 will like the item I_3 (with rating 4.82) and that the user U_5 will like the item I_8 (with rating 3.99). However, this technique also makes some predictions very hard to understand: the user U_6 will like the item I_{15}, and the user U_{11} will like the item I_{11}. As may be seen, these predictions do not fulfill Proposition 1: there are no users with similar tastes to those of user U_6 who have positively rated the item I_{15}; and there are no users with similar taste to those of user U_{11} who have positively rated the item I_{11}.

Like in the previous example, the user and item matrices do not provide information about the possible groups of users in the database. In Tables 11 and12 we describe respectively the user and item matrices that the classic matrix factorization outputs. As may be seen, these matrices are not intuitive to interpret and it is not possible to infer from them the three groups of users.

Table 8
Matrix associated to items according to our technique. Each term in this matrix indicates the probability that the users of a group like each item. As may be seen: users in group 1 like the items $\{I_1, I_2, I_3, I_4, I_5\}$; users in group 2 like the items $\{I_6, I_7, I_8, I_9, I_{10}\}$; and users in group 3 like the items $\{I_{11}, I_{12}, I_{13}, I_{14}, I_{15}\}$.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
F_1	0.9	0.9	0.9	0.9	0.9	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
F_2	0.5	0.5	0.5	0.5	0.5	0.9	0.9	0.9	0.9	0.9	0.5	0.5	0.5	0.5	0.5
F_3	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.9	0.9	0.9	0.9	0.9

Table 9
Second example of rating matrix.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
U_1	5	5	5	5	5	•	1	•	•	•	1	•	•	•	•
U_2	5	5	5	5	5	•	•	1	•	•	•	•	•	•	•
U_3	5	5	•	5	5	•	•	•	•	•	•	3	•	•	•
U_4	•	•	1	•	•	5	5	5	5	5	•	1	•	2	•
U_5	•	3	•	•	2	5	5	•	5	5	•	2	•	4	•
U_6	•	1	•	•	•	5	5	5	5	5	•	•	•	•	•
U_7	•	•	•	4	•	•	•	•	1	•	5	4	5	5	5
U_8	•	1	•	•	•	•	4	•	•	•	5	5	5	4	5
U_9	•	•	•	•	•	3	•	•	3	•	5	4	5	5	5
U_{10}	5	5	5	5	5	1	5	2	1	5	•	•	•	•	•
U_{11}	•	1	•	•	•	5	5	5	5	5	•	•	5	•	•

Table 10
Predictions of the ratings according to the classical matrix factorization.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
U_1	**4.96**	**4.90**	**5.01**	**4.96**	**4.95**	2.29	**1.16**	0.83	2.34	3.84	**1.16**	3.56	3.85	5.87	3.85
U_2	**4.94**	**5.03**	**4.98**	**4.91**	**5.00**	1.58	3.02	**1.21**	1.56	4.35	2.92	3.20	4.28	5.57	4.28
U_3	**5.04**	**4.97**	4.82	**5.05**	**4.80**	2.10	4.24	2.03	2.07	4.86	3.91	**3.13**	4.61	5.42	4.61
U_4	3.16	1.23	**1.11**	3.38	0.64	**4.95**	**6.79**	**4.99**	**4.96**	**5.00**	5.31	**1.13**	3.69	**2.19**	3.69
U_5	4.18	**2.97**	2.67	4.44	**2.24**	**4.90**	**5.02**	3.99	**4.97**	**4.95**	3.79	**2.14**	3.94	**3.84**	3.94
U_6	3.89	**1.16**	2.70	3.59	2.51	**4.99**	**4.99**	**4.99**	**4.95**	**4.96**	5.00	3.70	4.56	3.45	4.56
U_7	4.60	3.53	4.77	**4.15**	4.99	1.37	4.26	2.34	**1.23**	4.65	**4.94**	**4.02**	**4.94**	**4.92**	**4.94**
U_8	4.13	**1.19**	3.62	3.49	3.67	4.09	**4.05**	4.47	3.98	4.75	**4.99**	**4.87**	**4.96**	**4.03**	**4.96**
U_9	4.83	3.55	4.45	4.57	4.46	**3.08**	4.75	3.38	**3.01**	5.12	**4.91**	**3.97**	**5.02**	**4.94**	**5.02**
U_{10}	**4.97**	**4.95**	**4.96**	**4.87**	**5.06**	**1.22**	**4.88**	**1.90**	**1.13**	**4.94**	4.76	3.11	4.82	5.31	4.82
U_{11}	3.89	**1.16**	2.70	3.59	2.51	**4.99**	**4.99**	**4.99**	**4.95**	**4.96**	5.00	3.70	**4.56**	3.45	4.56

Table 11
Matrix associated to users according to the classic matrix factorization.

	F_1	F_2	F_3
U_1	−0.120950	−0.246555	1.914543
U_2	0.777214	−0.013833	1.085020
U_3	0.919117	−0.104120	0.562913
U_4	−0.029844	−0.992890	−1.711605
U_5	−0.293201	−1.057626	−0.328346
U_6	−0.781766	0.109562	−0.795306
U_7	0.901311	0.945192	0.109450
U_8	−0.769257	0.921102	−0.357261
U_9	0.351820	0.345726	−0.011454
U_{10}	1.472296	0.272105	0.232086
U_{11}	−0.781766	0.109562	−0.795306

When applying our technique on the rating matrix in Table 9, we obtain the predictions described in Table 13.

Like the classic matrix factorization, our technique also predicts that the user U_3 will like the item I_3 (with rating 5), and the user U_5 will like the item I_8. As we wished, our technique does not predict that user U_6 will like the item I_{15} (it does neither predict that user U_{11} will like the item I_{11}): the technique predicts that this rating will be 3 in order to minimize the average error.

Like the classic matrix factorization, our technique also provides a user matrix and an item matrix. However, unlike the classic matrix factorization, these matrices are very easy to understand and they are completely related to the groups of users in the database:

- In Table 14 we show the user matrix obtained with our technique. Each value $a_{u,k}$ of this matrix indicates the probability that the user u belongs to the group k. According to this matrix, we can easily derive the users for each group: group 1 with $\{U_1, U_2, U_3\}$; group 2 with $\{U_4, U_5, U_6\}$; and group 3 with $\{U_7, U_8, U_9\}$. As we expected above, the user U_{10} shares the taste of group 1 (with probability 0.86) as well as group 2 (with probability 0.13).
- In Table 15 we show the item matrix obtained with our technique. Each term $b_{i,k}$ of this matrix indicates the probability that users in group k like the item i. According to this matrix, we can easily derive the following: users in group 1 like the items $\{I_1, I_2, I_3, I_4, I_5\}$: users in group 2 like the items $\{I_6, I_7, I_8, I_9, I_{10}\}$; and users in group 3 like the items $\{I_{11}, I_{12}, I_{13}, I_{14}, I_{15}\}$.

Table 12
Matrix associated to items according to the classic matrix factorization.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
F_1	0.252	1.195	0.520	0.361	0.602	−1.337	0.648	−0.836	−1.365	0.132	0.429	−0.531	0.108	0.361	0.108
F_2	0.006	−0.634	0.580	−0.420	0.840	−0.780	−0.393	−0.014	−0.874	−0.111	0.650	1.222	0.500	0.160	0.500
F_3	0.526	1.197	0.991	0.553	1.052	−0.585	−1.435	−1.141	−0.555	−0.269	−1.244	0.429	−0.033	1.013	−0.033

Table 13
Predictions of the ratings according to our technique.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
U_1	**5**	**5**	**5**	**5**	**5**	2	**2**	2	2	4	**2**	3	3	3	3
U_2	**5**	**5**	**5**	**5**	**5**	2	2	**2**	2	4	2	3	3	3	3
U_3	**5**	**5**	5	**5**	**5**	2	2	2	2	4	2	**3**	3	3	3
U_4	3	2	**3**	3	3	**5**	**5**	**5**	**5**	**5**	3	**2**	4	**3**	3
U_5	3	**2**	3	3	**3**	**5**	**5**	5	**5**	**5**	3	**2**	4	**3**	3
U_6	3	**2**	3	3	3	**5**	**5**	**5**	**5**	**5**	3	2	4	3	3
U_7	3	2	3	**4**	3	3	4	3	**3**	3	**5**	**4**	**5**	**5**	**5**
U_8	3	**2**	3	4	3	3	**4**	3	3	3	**5**	**4**	**5**	**5**	**5**
U_9	3	2	3	4	3	**3**	4	3	**3**	3	**5**	**4**	**5**	**5**	**5**
U_{10}	**5**	**5**	**5**	**5**	**5**	**3**	**3**	**3**	**3**	**4**	2	3	3	3	3
U_{11}	3	**2**	3	3	3	**5**	**5**	**5**	**5**	**5**	3	2	**4**	3	3

Table 14
Matrix associated to users according to our technique. Each term of this matrix indicates the probability that a user belongs to a group. As may be seen, there are three groups: group 1 with $\{U_1, U_2, U_3\}$; group 2 with $\{U_4, U_5, U_6\}$; and group 3 with $\{U_7, U_8, U_9\}$. The user U_{10} belongs to both group 1 (with probability 0.86) and group 2 (with probability 0.13).

	F_1	F_2	F_3
U_1	**0.98**	0.01	0.01
U_2	**0.98**	0.01	0.01
U_3	**0.98**	0.01	0.01
U_4	0.01	**0.98**	0.01
U_5	0.01	**0.98**	0.01
U_6	0.01	**0.98**	0.01
U_7	0.01	0.01	**0.98**
U_8	0.01	0.01	**0.98**
U_9	0.01	0.01	**0.98**
U_{10}	**0.86**	**0.13**	0.01
U_{11}	0.01	**0.98**	0.01

4. Model

In this section we will present the probabilistic model of this paper. In Section 4.1 we will show the parameters, the input and output of our technique. In Section 4.2 we present the probabilistic model about how users rate items. In Section 4.3 we pose the mathematical problem related to our model. In Section 4.4 we discuss how to analyze the output of the algorithm. In Section 4.5, we describe the learning algorithm for this model. In Section 4.6, we finally present the effect of the parameters of our model in the output of our technique.

4.1. Overview of our technique

In this section we present a description of the algorithm as a black box. This algorithm will be useful for finding sets of users sharing the same tastes and predicting the probability that a user likes an item.

Input. The input of the algorithm is a matrix of ratings $\mathcal{M}$, since we are focusing on recommender systems based on collaborative filtering. We will use the following notation related to this rating matrix:

- N: Number of users in the ratings matrix.
- M: Number of items in the ratings matrix.
- $r_{u,i}$: Rating that user u has made over the item i. So that our algorithm can be used for any recommender system with different scales, we will consider the normalized ratings $r^*_{u,i}$ which lie within [0, 1]. In many recommender systems, users can rate items in a scale from 1 to 5 (where 1 indicates that the user does not like the item at all, and 5 indicates that user likes it very much). Consequently, for this kind of recommender systems, $r^*_{u,i}$ will be in {0.00, 0.25, 0.50, 0.75, 1.00} (since $r_{u,i} \in \{1, 2, 3, 4, 5\}$). We will use the notation $r_{u,i} = \bullet$ to indicate that user u has not rated the item i yet.

Parameters. Besides the rating matrix, our technique also takes into account some parameters fixed by the recommender system. Specifically we consider the following three parameters:

- $K \in \mathbb{N}$. This parameter indicates the number of groups of users that the algorithm is going to find out.

Table 15
Matrix associated to items according to our technique. Each term of this matrix indicates the probability that a user in a group likes each item. As may be seen: users in group 1 like the items $\{I_1, I_2, I_3, I_4, I_5\}$: users in group 2 like the items $\{I_6, I_7, I_8, I_9, I_{10}\}$; and users in group 3 like the items $\{I_{11}, I_{12}, I_{13}, I_{14}, I_{15}\}$.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
F_1	0.9	0.9	0.9	0.9	0.9	0.3	0.3	0.3	0.3	0.7	0.3	0.6	0.5	0.5	0.5
F_2	0.5	0.4	0.4	0.5	0.4	0.9	0.9	0.9	0.9	0.9	0.5	0.4	0.7	0.6	0.5
F_3	0.5	0.3	0.5	0.7	0.5	0.6	0.7	0.5	0.4	0.5	0.9	0.8	0.9	0.8	0.9

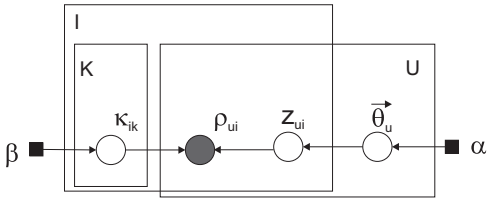

Fig. 1. Graphical model representation of our probabilistic approach.

- $\alpha \in (0, 1)$. This parameter is related to the possibility of obtaining overlapping groups of users sharing the same tastes. A value very close to 0 involves that users tend only to belong to one group. A greater value indicates that we are allowing that the user may be in more than one group.
- $\beta > 1$: This parameter is related to the amount of evidence that the algorithm requires to deduce that a group of users likes an item. The higher β, the more evidence the model requires to deduce that a group of users likes or dislikes an item.

Output. From the input and parameters, the algorithm outputs two matrices:

- The $N \times K$ matrix $(a_{u,k})$ associated to users. This matrix informs about the probability that user u belongs to the group k. In this way, for every user u, we have that:

$$\sum_{k=1}^{K} a_{u,k} = 1$$

- The $K \times M$ matrix $(b_{k,i})$ associated to items. This matrix informs about the the probability that users in the group k like the item i.

Calculations from the output matrices. From the output of the algorithm, we can immediately calculate important issues on the users and the items:

- Probability that the user u likes the item i. This measure $p_{u,i}$ is calculated in the following way:

$$p_{u,i} = \sum_{k=1\ldots K} a_{u,k} \cdot b_{k,i} \tag{1}$$

- Expected rating of the user u on the item i. Once we have calculated the probability that user u likes the item i, we can transform it to obtain the non-normalized rating this user would make of the item i. When the recommender system uses $\{1, \ldots, 5\}$ as the rating scale, this is obtained according to the following expression:

$$q_{u,i} = \begin{cases} 1 & if \quad 0 \le p_{u,i} < 0.2 \\ 2 & if \quad 0.2 \le p_{u,i} < 0.4 \\ 3 & if \quad 0.4 \le p_{u,i} < 0.6 \\ 4 & if \quad 0.6 \le p_{u,i} < 0.8 \\ 5 & if \quad 0.8 \le p_{u,i} \le 1 \end{cases} \tag{2}$$

4.2. The probabilistic model

Once we have presented the description of our algorithm as a black box, we will show our probabilistic model. In the Fig. 1, we show the Bayesian probabilistic model we are assuming: circles in the figure represent random variables; arrows between two variables indicate dependence between two variables. A gray circle indicates that the value of this random variables is observed. The small black squares represent the parameters of our model.

According to this figure, we can simulate the ratings that users make over the items by means of the following procedure:

- For each user u, we sample a K dimensional vector random variable $\vec{\phi}_u$ from a Dirichlet distribution:

$$\vec{\phi}_u \sim Dir(\underbrace{\alpha, \ldots, \alpha}_{K})$$

The vector $(\phi_{u,1}, \ldots, \phi_{u,K})$ represents the probability that a user belongs to each group. Dirichlet distribution is here chosen because it is typically used in variables whose support is the probability distribution of a categorical variable. That is to say, $\vec{\phi}_u$ does not take real or integer values like variables following typical distributions (normal distribution, binomial distribution, etc.), as well as, it is the prior conjugate distribution of the Categorical distribution used for modeling $z_{u,i}$ which is essential (as may be seen in the Appendix) for deriving the equations provided in Section 4.3.

- For each item i and each factor k, we sample a random variable $\kappa_{i,k}$ from a Beta distribution (which takes values from 0 to 1):

$$\kappa_{i,k} \sim Beta(\beta, \beta)$$

The value $\kappa_{i,k}$ represents the probability that a user in group k likes the item i. Beta distribution is here chosen because it is typically used in variables whose support is the probability distribution of a Binomial variable, as well as, it is the prior conjugate distribution of the Binomial distribution used for modeling $\rho_{u,i}$, which is essential (as may be seen in the Appendix) for deriving the equations provided in Section 4.3.

- For the user u and each item i such that $r_{u,i} \neq \bullet$, we sample the following random variables:
  - The random variable $z_{u,i}$ from a categorical distribution (which takes values in $\{1, \ldots, K\}$).

$$z_{u,i} \sim Cat(\vec{\phi}_u)$$

A value k in $z_{u,i}$ represents that the user u rates the item i as if the user belongs to the group k (since the support of $z_{u,i}$ is a finite set of values, $z_{u,i}$ must follow a categorical distribution).
  - The random variable $\rho_{u,i}$ from a Binomial distribution (which takes values from 0 to R). In our experiments we have set R to 4.

$$\rho_{u,i} \sim Bin(R, \kappa_{i,z_{u,i}})$$

We obtain the normalized rating from the value $\rho_{u,i}$ according to the expression:

$$r^*_{u,i} = \frac{\rho_{u,i}}{R}$$

Unlike other probabilistic models [5,18], we use a Binomial distribution to model the ratings because of the following:

- The Binomial distribution is discrete and the ratings in recommender systems are also discrete. In other probabilistic models (like [5,18]) the variable modeling the ratings follow a Normal distribution, which is not a discrete distribution.
- Using categorical distribution (proposed in [18]) does not take into account the order relation of the rating values.
- The Binomial distribution may plausibly model the way that user evaluates the items: a user evaluates R features of the items and the rating indicates the number of these features that the user likes.
- There is only one unknown parameter for this distribution: the probability that the user likes a feature of the item. This reduces the possibility of overfitting in our model.
- The prior distribution of the Binomial distribution is the well-known Beta distribution (which the random variable $\kappa_{i,k}$ follows). This distribution in Bayesian framework represents the confidence (prior knowledge) that a given user likes or not an item. Unlike Gaussian distribution (which is not bounded), the Beta distribution takes values within [0, 1].

Following this procedure, we can simulate samples of plausible ratings. About this procedure, we simply observe the value of the random variables $\rho_{u,i} = r^*_{u,i} \cdot R$ (gray circles in Fig. 1). The rest of variables (white circles in Fig. 1) are unknown for us. The important issue about the model lies in finding out the distribution of these unknown random variables (which are sometimes called latent variables), that is to say, the conditional probability distribution of the unknown variables considering that we observe and know the variable $\rho_{u,i}$. The task of finding out these distributions will be discussed in the next section.

4.3. Inference in the model: variational inference

Once we have considered the probabilistic model discussed in Section 4.2, we would like to determine the conditional probability distribution of the unknown random variables (white circles in Fig. 1) considering that we know the rating matrix $\mathcal{M}$ (that is to say, the value of the random variables $\rho_{u,i}$).

Like almost all probabilistic graphical models, the probabilistic model considered does not allow to computationally calculate the exact conditional distributions in an analytical way. Like in [2,19], we have adopted the variational inference technique (also called Variational Bayes) [1] to calculate an approach distribution of the real posterior distribution that can be efficiently calculated. This technique can be immediately derived since our model has a conjugate-exponential structure: variables follow distributions in the exponential family of distributions; and the distribution of any variable conditioned by the others follow the same distribution without conditioning. Thanks to this, we can use the framework [7], and approximate the real posterior distribution $p(\vec{\phi}_u, \kappa_{i,k}, z_{u,i}|\rho_{u,i})$ by a distribution $q(\vec{\phi}_u, \kappa_{i,k}, z_{u,i})$ (the proof for this technique can be seen in detail in the Appendix):

$$q(\vec{\phi}_u, \kappa_{i,k}, z_{u,i}) = \prod_{u=1}^{N} q_{\vec{\phi}_u}(\vec{\phi}_u) \prod_{i=1}^{M} \prod_{k=1}^{K} q_{\kappa_{i,k}}(\kappa_{i,k}) \prod_{r_{u,i}\neq\bullet} q_{z_{u,i}}(z_{u,i})$$

where:

- The distribution $q_{\vec{\phi}_u}(\vec{\phi}_u)$, $q_{\kappa_{i,k}}(\kappa_{i,k})$ and $q_{z_{u,i}}(z_{u,i})$ have the following distributions according to the framework [7]:
  - The approximated conditional probability distribution of $\vec{\phi}_u$ knowing the rating matrix $\mathcal{M}$, also follows a Dirichlet distribution:
    $$q_{\vec{\phi}_u}(\vec{\phi}_u) \sim Dir(\gamma_{u,1}, \ldots \gamma_{u,K}) \tag{3}$$
    where $\gamma_{u,1}, \ldots \gamma_{u,k}$ are parameters to be learned, as we will next see.
  - The approximated conditional probability distribution of $\kappa_{i,k}$ knowing the rating matrix $\mathcal{M}$ also follows a Beta distribution:
    $$q_{\kappa_{i,k}}(\kappa_{i,k}) \sim Beta(\epsilon^+_{i,k}, \epsilon^-_{i,k}) \tag{4}$$
    where $\epsilon^+_{i,k}, \epsilon^-_{i,k}$ are parameters to be learned, as we will next see.
  - The conditional probability distribution of $z_{u,i}$ knowing the rating matrix $\mathcal{M}$ also follows a categorical distribution:
    $$q_{z_{u,i}}(z_{u,i}) \sim Cat(\lambda_{u,i,1}, \ldots, \lambda_{u,i,K}) \tag{5}$$
    where $\lambda_{u,i,k}$ are parameters to be learned, as we will next see. Since $q_{z_{u,i}}(z_{u,i})$ has a Categorical distribution, these parameters must fulfill:
    $$\lambda_{u,i,1} + \cdots + \lambda_{u,i,K} = 1$$
- The previously described parameters $\gamma_{u,k}, \epsilon^+_{i,k}, \epsilon^-_{i,k}, \lambda_{u,i,k}$ must fulfill the following conditions, according to the framework [7]:

$$\gamma_{u,k} = \alpha + \sum_{\{i|r_{u,i}\neq\bullet\}} \lambda_{u,i,k} \tag{6}$$

$$\epsilon^+_{i,k} = \beta + \sum_{\{u|r_{u,i}\neq\bullet\}} \lambda_{u,i,k} \cdot r^+_{u,i} \tag{7}$$

$$\epsilon^-_{i,k} = \beta + \sum_{\{u|r_{u,i}\neq\bullet\}} \lambda_{u,i,k} \cdot r^-_{u,i} \tag{8}$$

$$\begin{aligned} \lambda'_{u,i,k} &= \exp\Big(\Psi(\gamma_{u,k}) + r^+_{u,i} \cdot \Psi(\epsilon^+_{i,k}) + r^-_{u,i} \cdot \Psi(\epsilon^-_{i,k}) \\ &\quad - R \cdot \Psi(\epsilon^+_{i,k} + \epsilon^-_{i,k})\Big) \\ \lambda_{u,i,k} &= \frac{\lambda'_{u,i,k}}{\lambda'_{u,i,1} + \cdots + \lambda'_{u,i,K}} \end{aligned} \tag{9}$$

where:

- Ψ is the digamma function defined as the logarithmic derivative of the gamma function:
  $$\Psi(x) = (\ln \Gamma(x))' = \frac{\Gamma'(x)}{\Gamma(x)}$$
- $r^+_{u,i} = \rho_{u,i} = R \cdot r^*_{u,i}$
- $r^-_{u,i} = R - \rho_{u,i} = R \cdot (1 - r^*_{u,i})$

4.4. Calculations from the model

Once the algorithm discussed in Section 4.3 have been run to determine the value of the free parameters $\{\gamma_{u,k}, \epsilon^+_{i,k}, \epsilon^-_{i,k}, \lambda_{u,i,k}\}$, we can immediately obtain important calculations about the tastes of users:

- About finding out groups of users sharing the same taste:
  - The probability $a_{u,k}$ that the user u belongs to the group k. This is the expected value of the conditional distribution of (3). In this way, the value $a_{u,k}$ is:
    $$a_{u,k} = P(u \text{ belongs to group } k) = \frac{\gamma_{u,k}}{\sum_{j=1\ldots K} \gamma_{u,j}} \tag{10}$$
    The value $a_{u,k}$ can be also considered as a soft membership function of the group k for the user u. As may be easily seen, for every user u, we have that:
    $$\sum_{k=1}^{K} a_{u,k} = 1$$
  - The probability $b_{k,i}$ that users in group k like the item i. This is the expected value of the conditional distribution of (4). In this way, the value $b_{k,i}$ is:
    $$b_{k,i} = P(u \text{ likes } i | u \text{ belongs to group } k) = \frac{\epsilon^+_{i,k}}{\epsilon^+_{i,k} + \epsilon^-_{i,k}} \tag{11}$$
- About predictions of the tastes of users.
  First of all, we need to obtain the conditional probability distribution $\rho_{u,i}$ from the distributions learned in Section 4.3. This random variable takes values from 0 to $R-1$ according to the number of different rating values the recommender system supports. In order to find values regardless of R, we are going to consider the random variable $\frac{1}{R}\rho_{u,i}$ which takes values in [0, 1] (regardless of the value R), representing the probability of liking an item. Indeed we are going to calculate the expected value of this random variable $\frac{1}{R}\rho_{u,i}$:
  $$p_{u,i} = \mathbf{E}\left(\frac{1}{R}\rho_{u,i}|\mathcal{M}\right) = \frac{1}{R}\mathbf{E}(\rho_{u,i}|\mathcal{M}) = \sum_{k=1}^{K} a_{u,k} b_{k,i}$$

Table 16
Relative frequency of the significant components in the user vectors.

Number of significant components	$\alpha = 0.3(\%)$	$\alpha = 0.5(\%)$	$\alpha = 0.8(\%)$
1	36	17	9
2	50	49	31
3	13	26	27
4	1	8	20
5	0	0	10
6	0	0	2

4.5. Algorithm

As may be seen in Eqs. (6)–(9), the calculation of one of the free parameters $\{\gamma_{u,k}, \epsilon^{+}_{i,k}, \epsilon^{-}_{i,k}, \lambda_{u,i,k}\}$ involves knowing the rest of them.

In order to find out all these parameters, we will use a coordinate ascent algorithm typically used in variational inference technique for calculating these parameters: we will change the values of these free parameters iterating from these Eqs. (6)–(9) until convergence (which is guaranteed by [21]) is reached.

- Input: $\mathcal{M} = (r^{*}_{u,i})$, α, β
- Output: $(\gamma_{u,k})$, $(\epsilon^{-}_{i,k})$, $(\epsilon^{+}_{i,k})$
- Steps:
  - Initialize randomly $\gamma_{u,k}$.
  - Initialize randomly $\epsilon^{+}_{i,k}$.
  - Initialize randomly $\epsilon^{-}_{i,k}$.
  - Repeat until changes are not significant
    - For each user u:
    - For each item i rated by the user u in the training set:
    - For each factor k: update $\lambda_{u,i,k}$ according to Eq. (9).
    - For each user u:
    - For each factor k: update $\gamma_{u,k}$ according to Eq. (6).
    - For each item i rated by the user u in the training set:
    - For each factor k: update $\epsilon^{+}_{i,k}$ according to Eq. (7).
    - For each factor k: update $\epsilon^{-}_{i,k}$ according to Eq. (8).
- Output $a_{u,k}$ according to Eq. (10).
- Output $b_{k,i}$ according to Eq. (11).

4.6. Effect of the parameters in our model

In this section, we will analyze the effect the parameters have on our model. As may be seen, our model considers the following parameters:

Parameter K. This parameter informs about the number of groups of users that we are going to consider for predicting the ratings.

Parameter $\alpha \in (0, 1)$. This parameter relates to the degree of sparsity of user vectors, $\vec{a}_u$, we wish in our model: the lower the parameter α, the more sparsity the vectors $\vec{a}_u$ have. In order to show the influence of this parameter in the sparsity of these vectors, we have trained our model with the database MovieLens 1M and with different values of α ($K = 6$, $\beta = 5$). In Table 16 we show the distribution of the number of significant components (components with value greater than 0.08) in the user vectors. As may be seen, while the most user vectors have just one (36%) or two components (50%) when $\alpha = 0.3$, there are more user vectors with three (27%) or more significant components when $\alpha = 0.8$.
As we will see in Section 5, sparsity is interesting not only because of the efficiency in calculating predictions according to Eq. (1), but also because this quality allows us to get simpler models and understand better the predictions of our model. Obviously, a rather small parameter α involves a very simple model that may predict worse than a greater α. The adequate parameter α may be selected by cross validation and taken into account the sparsity degree of user vectors we wish in our model.

Table 17
Percentage of predictions with value 3.

β	% predictions with value 3
$\beta = 5$	27.93
$\beta = 10$	29.20
$\beta = 15$	30.36
$\beta = 20$	31.14

Parameter $\beta > 1$. This parameter relates to the degree of evidence that our model needs for inferring that a user likes or dislikes an item: the higher β, the more evidence our model needs to infer that a user likes or dislikes an item. In order to show the influence of this parameter in the predictions, we have trained our model with the database MovieLens 1M with different values of β. In Table 17 we show the percentage of predictions with value 3 (which represents the indifference rating value). As may be seen, the higher β, the more ratings with value 3 the model predicts.
Like the parameter α, the parameter β may be also set by means of cross validation.

4.7. Model performance

Here we provide a theoretical section analyzing the complexity of the proposed model in terms of both computation and storage when making recommendations. Both in classical matrix factorization and in our model, predictions of the user's tastes are calculated as follows:

$$p_{u,i} = \sum_{k=1\ldots K} a_{u,k} \cdot b_{k,i} \tag{12}$$

Although the previous formula can be calculated immediately for a particular user u and item i, the recommender system requires to calculate it for each user and each item in order to find the best recommendations for each user. Consequently, if the number of items or users is high, it may be interesting to find a quick way of calculating $p_{u,i}$ in order to reduce the burden of the computation of the Recommender System. Here we provide a way which takes advantage of the characteristics of our model.

The recommender system recommends each user u those items i with the highest values in $p_{u,i}$ in such way that $p_{u,i} > S$. The value S is a threshold value indicating when an item i can be recommendable to user u.

- For classical matrix factorization, the value $p_{u,i}$ represents the prediction of the rating that user u would make on item i if user u consumed the item i. In this way, in recommender systems which use the typical range of ratings $\{1, \ldots, 5\}$, the value S is usually set to 4. For the general case of range of ratings, considering the normalized rating values within the interval [0,1], the value S would be set to 0.8.
- Unlike the classical matrix factorization, both the values $p_{u,i}$ and S have a probabilistic meaning: an item i is recommendable to user u if the estimated probability $p_{u,i}$ that the user u will like the item i is higher than S.

As we have stated above, the Eq. (12) can be used in the classical matrix factorization as well as in our approach. However, we can make use of an alternative way to check whether an item i is recommendable for user u or not, since, unlike classical matrix factorization, we have in our approach that: $0 \leq a_{u,k} \leq 1$; $0 \leq b_{k,i} \leq 1$; $\sum_{k=1}^{K} a_{u,k} = 1$; and $\vec{a}_u$ is typically very sparse.

For our approach, the following algorithm outputs a Boolean value indicating if an item i is recommendable to a user u:

- Input: A user u; an item i; the threshold value S indicating whether the item i is recommendable to user u or not.
- Output: A Boolean value indicating whether the item i is recommendable to user u or not.
- Steps:
  1. $sum \leftarrow 0$
  2. $p \leftarrow 0$
  3. For k = 1 to K
     3.1 $sum \leftarrow sum + a_{u,k}$
     3.2 $p \leftarrow p + a_{u,k} \cdot b_{k,i}$
     3.3 if $p > S$ then Output 'Recommendable'.
     3.4 if $p + (1 - sum) < S$ then Output 'Not Recommendable'.

Like in classical matrix factorization, this algorithm calculates $p_{u,\,i}$ through the partial sums of Eq. (12) stored in the variable p of the algorithm. However, this algorithm differs from the one of classical matrix factorization in steps 3.3 and 3.4:

- Step 3.3. Since $a_{u,\,k} \geq 0$ and $b_{u,\,k} \geq 0$, the partial sum of Eq. (12) increases with the value k (unlike the classical matrix factorization where $a_{u,\,k}$ or $b_{k,\,i}$ may take negative values). Consequently, when a partial sum of Eq. (12) is greater than S (that is to say, $p > S$), we can state that $p_{u,\,i} > S$ (that is to say, the item i is recommendable to user u).
- Step 3.4. Since $a_{u,\,k} \geq 0$, and $b_{k,\,i} \leq 1$, we obtain an immediate upper bound of the value $p_{u,\,i}$.
  $p_{u,i} = \underbrace{a_{u,1} \cdot b_{1,i} + a_{u,2} \cdot b_{2,i} + \cdots + a_{u,k} \ldots b_{k,i}}_{p} + a_{u,k+1} \cdot b_{k+1,i}$
  $+ \cdots + a_{u,K} \ldots b_{K,i} = p + a_{u,k+1} \cdot b_{k+1,i} + \cdots + a_{u,K} \cdot b_{K,i} \leq p$
  $+ a_{u,k+1} \cdot 1 + \cdots + a_{u,K} \cdot 1 = p + (1 - \sum_{t=1}^{k} a_{u,k}) = p + (1 - sum)$
  Therefore, if $p + (1 - sum) < S$, we can state that $p_{u,\,i} < S$ (the item i is not recommendable to user u).

As may be seen, this algorithm runs always faster than the one for the classical matrix factorization, since we do not need here to calculate all the terms of the Eq. (12).

Besides, we can improve significantly the algorithm for the usual case that the vector a_u is sparse (as we have seen in Section 4.6 this can be controlled by the parameter α). In this case, few components $a_{u,\,k}$ take high values while most components of $a_{u,\,k}$ take values close to 0 (so that the product $a_{u,\,k} \cdot b_{k,\,i}$ can be regarded as 0 for practical issues). In this case, for each user u we can store two small lists of the same size:

- $k_{u,\,n}$: The factor k which is the n highest component of $\vec{a}_u$.
- $a'_{u,n}$: The n highest component of $\vec{a}_u$. That is to say, $a'_{u,n} = a_{u,k_{u,n}}$

Since a_u is sparse, we will store only the significant components of $a_{u,\,k}$, and therefore the size of these lists, (denoted by N_u) will be $N_u < < K$). In this case, the algorithm for determining if an item is recommendable to a user is:

- Input: A user u; An item i; the threshold value S indicating whether the item i is recommendable to user u or not.
- Output: A Boolean value indicating whether the item i is recommendable to user u or not.
- Steps:
  1. $sum \leftarrow 0$
  2. $p \leftarrow 0$
  3. For n = 1 to N_u
     3.1 $sum \leftarrow sum + a'_{u,n}$
     3.2 $k \leftarrow k_{u,\,n}$
     3.3 $p \leftarrow p + a'_{u,n} \cdot b_{k,i}$
     3.4 if $p > S$ then Output 'Recommendable'.
     3.5 if $p + (1 - sum) < S$ then Output 'Not Recommendable'.

As may be seen, this algorithm is very similar to the previous one. Nevertheless, it takes advantage of the fact that the vector $\vec{a}_u$ is sparse. In this way, we only need to store $N_u \ll K$ components (instead of K). Besides, the algorithm iterates at most N_u times instead of K. Consequently, this algorithm involves a reduction of $\frac{N_u}{K}$ over the previous one in terms of cost of computation and storage. Besides, since this algorithm involves that $a'_{u,n}$ is high in the first iterations, p and sum will take high values in these first iterations and the conditions in lines 3.4 and 3.5 can verify earlier.

5. Understanding the recommendations

Unlike the classical matrix factorization, our model allows to understand better the predictions, thanks to the following features of the user and item vectors: the components of these vectors are non-negative; the user vectors tend to be sparse (very few components take high value, and most of them take almost null values).

According to Eq. (1), the probability that a user u likes the item i is calculated in our model as follows:

$$p_{u,i} = \vec{a}_u \cdot \vec{b}_i = \sum_{k=1}^{K} a_{u,k} b_{k,i}$$

As described in [8], the fact that the components $a_{u,\,k}$ and $b_{k,\,i}$ are non-negative involves, unlike the classical matrix factorization, that terms cannot be canceled. In this way, each term influences positively the value of the prediction. Besides, each component $a_{u,\,k}$ and $b_{k,\,i}$ has an understandable probabilistic meaning. Indeed, according to our model, we are considering that there are K main types of users' tastes, and every user has an additive combination of the tastes of these K users.

In addition to this, user vectors are usually sparse, which allows us to understand and justify the predictions of our model in the same way as the K-NN algorithm: if a prediction $p_{u,\,i}$ is high then there are other users with the same taste as u that have positively rated the item i.

If user vectors are sparse enough, then we can conclude that Proposition 1 holds in our algorithm: if we have predicted that $p_{u,\,i}$ is very high (close to 1), then we have that there is a $k \in \{1 \ldots K\}$ such that $a_{u,\,k}$ (because of the sparsity of the user vectors) and $b_{k,\,i}$ (because the prediction is high) are close to 1. According to the meaning of these vectors, we have that:

- The user u very likely belongs to the group k of users because $a_{u,\,k}$ is close to 1 (that is to say, the user u shares the same tastes as the users in group k).
- Many users in group k have positively rated the item i because $b_{k,\,i}$ is close to 1.

Consequently, we have that there are users with the same taste as user u (because they belong to the same group) who have rated positively the item i. As may be seen, this kind of reasoning is the one used in the K-NN algorithm for understanding the recommendations provided.

6. Evaluation

In this section, we will analyze the accuracy of the predictions and recommendations of our technique [5,6]. We will compare our technique with other techniques, and we have specially focused on the classical matrix factorization: the technique that provides better results when only considering the ratings matrix. As we have stated above, this technique has been extended in order to consider other factors like implicit information about preference of users [15], hybrid systems using content-based information of items [13] or social information [14]. All these factors could be also integrated and taken into account as future work in our model. We

Table 18
Accuracy of the predictions.

Technique	MovieLens 1M			MovieLens 10M			NetFlix		
	MAE	CMAE	0–1 Loss	MAE	CMAE	0–1 Loss	MAE	CMAE	0–1 Loss
KNN-Pearson correlation	0.823	0.721	0.423	0.842	0.743	0.434	0.835	0.735	0.469
KNN JMSD [11]	0.743	0.695	0.396	0.753	0.701	0.405	0.748	0.695	0.421
NMF [8]	1.243	1.106	0.463	1.356	1.234	0.487	1.286	1.106	0.438
PLSA-CF[18]	0.796	0.764	0.393	0.823	0.785	0.402	0.805	0.764	0.469
Classical matrix factorization [5,6]	0.684	0.574	0.384	0.698	0.586	0.396	0.692	0.574	0.375
Our approach	0.664	0.526	0.270	0.676	0.542	0.284	0.666	0.536	0.248

have used the database MovieLens 1M, MovieLens 10M and NetFlix for evaluating our technique, because they are established references in the research literature. We have divided this database of ratings into two sets: 80% of ratings in the training set, E, used by the learning algorithm described in Section 4.5; and 20% in the test set, T, used to measure and compare the accuracy of the technique. We have repeated this procedure 20 times and calculated the average of the quality metrics.

The values for the parameters in [6] have been selected so that they provide the best results: $\lambda = 0.055$ and $K = 15$; In relation to our technique, we have chosen $\alpha = 0.8$; $\beta = 5$ and $K = 6$.

6.1. Accuracy in predictions

In this section we will study the accuracy of the predictions of our technique through Mean Absolute Error (MAE). This measure calculates the average absolute error in the training data. That is to say:

$$\text{MAE} = \frac{\sum_{(u,i)\in T} |r_{u,i} - q_{u,i}|}{|T|}$$

Although the MAE in techniques based on the K-NN strongly depends on the similarity metric [9–12,33], this technique does not manage to improve MAE under 0.75. On the other hand, applying the non-negative factorization technique over the rating matrix [8] provides in our experiments a value even worse than K-NN (greater than 0.8).

In Table 18, we compare the MAE for our approach and other techniques. As may be seen, the MAE of our approach is even lower than the one obtained with classical Matrix factorization. Besides, our approach has the advantage of providing understandable user and item vectors with which we can easily justify the recommendations.

In the next experiment, we have only considered the relevant errors in the ratings: the relevant error lies in those items that users are predicted to like ($q_{u,i} \geq 4$) and in those items that users really like ($r_{u,i} \geq 4$). For a recommender system, it is an insignificant error if the system predicts $q_{u,i} = 1$ when $r_{u,i} = 2$ (or $q_{u,i} = 2$ when $r_{u,i} = 1$), but it is a quite important error if the system predicts that $q_{u,i} = 4$ when $r_{u,i} = 3$ (or $q_{u,i} = 3$ when $r_{u,i} = 4$). In order to take into account this issue, we have defined the following metric (called Constrained Mean Absolute Error):

$$\text{CMAE} = \frac{\sum_{(u,i)\in T'} |r_{u,i} - q_{u,i}|}{|T'|}$$

where $T' = \{(u,i) \in T | q_{u,i} \geq 4 \vee r_{u,i} \geq 4\}$

For this metric, our approach is significantly better than the classical matrix factorization (see Table 18). This means that our technique predicts the relevant ratings far better than the classical matrix factorization.

Besides, we have studied the accuracy of the predictions according to the 0–1 loss function: we distinguish the items according to the predictions between recommendable (predictions greater or equal than 4) and non-recommendable. The 0-1-Loss metric studies the amount of mistakes in recommendable and non-recommendable predictions.

Table 19
Quality of recommendations. Precision measured in percentage (%).

DataBase	Algorithm	Number of recommendations			
		5	10	15	20
MovieLens 1M	Classical matrix factorization	4.943	4.976	4.852	4.666
	Our approach	11.159	9.471	8.553	7.872
MovieLens 10M	Classical matrix factorization	5.336	5.135	4.936	4.874
	Our approach	11.468	9.865	8.685	7.947
NetFlix	Classical matrix factorization	5.165	5.047	4.825	4.752
	Our approach	11.268	9.654	8.254	7.356

The 0-1-Loss function is calculated according to the following expression:

$$\text{0-1-Loss} = \frac{\sum_{(u,i)\in T} h(r_{u,i}, q_{u,i})}{|T|}$$

$$\text{where } h(r_{u,i}, q_{u,i}) = \begin{cases} 0 & \text{if } r_{u,i} \geq 4 \wedge q_{u,i} \geq 4 \\ 0 & \text{if } r_{u,i} < 4 \wedge q_{u,i} < 4 \\ 1 & \text{otherwise} \end{cases}$$

According to the results in Table 18, we have that our approach significantly improves the accuracy in the 0-1-Loss.

6.2. Accuracy in recommendations

In the previous section, we have studied the quality of the predictions. Here, we will study the quality of the recommendations through the precision and recall measures used in recommender systems. First, we will remind the formal definition of these metrics.

Let N be the number of recommendations provided for each user.

Let R be the set of the N items not rated by each user u in the training set ($r_{u,i} = \bullet$) that have been predicted with the highest values. The set R represents the set of the N recommendations for each user provided by the recommender system.

Let S be the set of the items rated by each user in the test set with high value (greater than 4). The set S represents the set of the items that should have been recommended to the users (we know this because $r_{u,i} \geq 4$ in the test set).

The recall measure is defined as the percentage of items correctly recommended over the number of items that should be recommended. That is to say:

$$\text{Recall} = \frac{|R \cap S|}{|S|} \cdot 100$$

The precision measure is defined as the percentage of items correctly recommended over the number of items recommended. That is to say:

$$\text{Precision} = \frac{|R \cap S|}{|R|} \cdot 100$$

Table 20
Quality of recommendations. Recall measured in percentage (%).

DataBase	Algorithm	Number of recommendations			
		5	10	15	20
MovieLens 1M	Classical matrix factorization	1.297	2.614	3.821	4.899
	Our approach	2.931	4.975	6.739	8.269
MovieLens 10M	Classical matrix factorization	1.467	2.754	3.975	4.974
	Our approach	3.068	5.064	6.846	8.356
NetFlix	Classical matrix factorization	1.365	2.456	3.794	4.567
	Our approach	3.124	4.953	6.843	8.462

In Tables 19 and 20 we show the precision and recall for different number of recommendations in our algorithm and the classical matrix factorization. As may be seen, our technique provides much better results in precision and recall than classical matrix factorization. We think that this improvement is due to the fact that our technique, unlike classical matrix factorization, tends to predict with indifference (value 3) those ratings for which the algorithm has not found out enough evidence (see Section 3).

7. Conclusions

In this paper we have presented a novel technique for recommender systems based on collaborative filtering. Like classical matrix factorization, our technique associates a vector of K component to each user and each item. However, unlike the classical matrix factorization, the components of these vectors lie within [0, 1] with an understandable probabilistic meaning which provides important advantages over the classical matrix factorization [5,6]: they allow us to identify groups of users sharing the same tastes, and justify the recommendations in the same way as the K-NN algorithm. Besides, our technique is very competitive with respect to the classical matrix factorization in terms of accuracy in predictions and recommendations.

Acknowledgment

This research was supported by the Spanish Ministerio de Economía y Competitividad TIN2012-32682 project.

Appendix

Given the rating matrix $\mathcal{M}$, we need to calculate the posterior distribution of user vectors $\vec{a}_u$ and item vectors $\vec{b}_i$. Our derivation of the variational algorithm makes use of general results about the class of conditionally conjugate models [1].

Variational inference. Variational inference fits the variational parameters to minimize their KL divergence to the posterior.

$$\min_q KL(q(\vec{\phi},\kappa,z)||p(\vec{\phi},\kappa,z|\rho))$$

where $q(\vec{\phi},\kappa,z)$ has the following expression:

$$q(\vec{\phi},\kappa,z)=\prod_{u=1}^{N} q_{\vec{\phi}_u}(\vec{\phi}_u)\prod_{i=1}^{M}\prod_{k=1}^{K} q_{\kappa_{i,k}}(\kappa_{i,k})\prod_{r_{u,i}\neq\bullet} q_{z_{u,i}}(z_{u,i})$$

According to our model described in Section 4.2, we have that:

$$p(\vec{\phi},\kappa,z,\rho)=\prod_{u=1}^{N}\prod_{i=1}^{M}\prod_{k=1}^{K}\prod_{r_{u,i}\neq\bullet} p(\rho_{u,i}|\kappa_{i,k},z_{u,i})\cdot p(z_{u,i}|\vec{\phi}_u)\cdot p(\vec{\phi}_u|\alpha)\cdot p(\kappa_{i,k}|\beta)$$

We define the following:

$$\lambda_{u,i,k}=q_z(z_{u,i}=k)$$

As may be seen in section 10.1.1 in [7], we can find an expression for each partial factor of q:

- Derivation of distribution (3) and Eq. (6).

$$\begin{aligned} q_{\vec{\phi}_u}(\vec{\phi}_u) &\propto \exp\{\mathbf{E}_{q_\kappa,q_z}[\ln p(\vec{\phi}_u,\kappa,z,\rho)]\}\\ &= \exp\{\mathbf{E}_{q_\kappa,q_z}[\ln\{p(\vec{\phi}_u|\kappa,z,\rho)p(\kappa,z,\rho)\}]\}\\ &= \exp\{\mathbf{E}_{q_\kappa,q_z}[\ln p(\vec{\phi}_u|\kappa,z,\rho)]+\mathbf{E}_{q_\kappa,q_z}[\ln p(\kappa,z,\rho)]\}\\ &\propto \exp\{\mathbf{E}_{q_\kappa,q_z}[\ln p(\vec{\phi}_u|\kappa,z,\rho)]\}\\ &\quad + constant \propto \exp\{\mathbf{E}_{q_z}[\ln p(\vec{\phi}_u|z_{u,i_1}\dots z_{u,i_M})]\}\end{aligned}$$

Since Dirichlet distribution is the conjugate distribution of the categorical distribution, we have that $\vec{\phi}_u|z_{u,i_1}\dots z_{u,i_M}$ follows a Dirichlet distribution (ϕ_u follows a Dirichlet distribution and $z_{u,i}$ are categorical distributions) with the following parameters:

$$\vec{\phi}_u|z_{u,i_1}\dots z_{u,i_M}\sim Dir\left(\alpha+\sum_{i,r_{u,i}\neq\bullet} I(z_{u,i}=1),\dots,\alpha+\sum_{i,r_{u,i}\neq\bullet} I(z_{u,i}=K)\right)$$

where I_C is an indicator function that outputs 1 if the condition C is true, and outputs 0 otherwise.

$$\begin{aligned} q_{\vec{\phi}_u}(\vec{\phi}_u) &\propto \exp\{\mathbf{E}_{q_z}[\ln p(\vec{\phi}_u|z_{u,i_1}\dots z_{u,i_M})]\}\\ &= \exp\left\{\mathbf{E}_{q_z}\left[\ln\left(\frac{\prod_{k=1}^{K}\Gamma(\alpha+\sum_{i,r_{u,i}\neq\bullet} I(z_{u,i}=k))}{\sum_{k=1}^{K}\Gamma(\alpha+\sum_{i,r_{u,i}\neq\bullet} I(z_{u,i}=k))}\right)\right.\right.\\ &\quad \left.\left.\times\prod_{k=1}^{K}\phi_{u,k}^{\alpha+\sum_{i,r_{u,i}\neq\bullet} I(z_{u,i}=k)-1}\right)\right]\Bigg\}\\ &\propto \exp\left\{\mathbf{E}_{q_z}\left[\ln\prod_{k=1}^{K}\phi_{u,k}^{\alpha+\sum_{i,r_{u,i}\neq\bullet} I(z_{u,i}=k)-1}\right]\right\}\\ &= \exp\left\{\mathbf{E}_{q_z}\left[\sum_{k=1}^{K}\left(\alpha+\sum_{i,r_{u,i}\neq\bullet} I(z_{u,i}=k)-1\right)\ln\phi_{u,k}\right]\right\}\\ &= \exp\left\{\sum_{k=1}^{K}\left(\alpha+\sum_{i,r_{u,i}\neq\bullet}\mathbf{E}_{q_z}[I(z_{u,i}=k)]-1\right)\ln\phi_{u,k}\right\}\\ &= \exp\left\{\sum_{k=1}^{K}\left(\alpha+\sum_{i,r_{u,i}\neq\bullet} q_z(z_{u,i}=k)-1\right)\ln\phi_{u,k}\right\}\end{aligned}$$

According to the following definitions:

- $\lambda_{u,i,k}=q_z(z_{u,i}=k)$
- $\gamma_{u,k}=\alpha+\sum_{i,r_{u,i}\neq\bullet}\lambda_{u,i,k}$

we have that:

$$\begin{aligned} q_{\vec{\phi}_u}(\vec{\phi}_u) &\propto \exp\left\{\sum_{k=1}^{K}\left(\alpha+\sum_{i,r_{u,i}\neq\bullet}\lambda_{u,i,k}-1\right)\ln\phi_{u,k}\right\}\\ &= \exp\left\{\sum_{k=1}^{K}(\gamma_{u,k}-1)\ln\phi_{u,k}\right\}=\prod_{k=1}^{K}(\phi_{u,k})^{\gamma_{u,k}-1}\end{aligned}$$

Consequently, we have that $q_{\vec{\phi}}$, the approximated conditional probability distribution of $\vec{\phi}_u$ knowing the rating matrix, also follows a Dirichlet distribution:

$$q_{\vec{\phi}_u}\sim Dir(\gamma_{u,1},\dots,\gamma_{u,k})$$

- Derivation of distribution (4), and Eqs. (7) and (8).

$$\begin{aligned} q_\kappa(\kappa_{i,k}) &\propto \exp\{\mathbf{E}_{q_{\vec{\phi}},q_z}[\ln p(\vec{\phi},\kappa_{i,k},z,\rho)]\} \\ &= \exp\{\mathbf{E}_{q_{\vec{\phi}},q_z}[\ln\{p(\kappa_{i,k}|\vec{\phi},z,\rho)\cdot p(\vec{\phi},z,\rho)\}]\} \\ &= \exp\{\mathbf{E}_{q_{\vec{\phi}},q_z}[\ln p(\kappa_{i,k}|\vec{\phi},z,\rho)] + \mathbf{E}_{q_{\vec{\phi}},q_z}[\ln p(\vec{\phi},z,\rho)]\} \\ &\propto \exp\{\mathbf{E}_{q_{\vec{\phi}},q_z}[\ln p(\kappa_{i,k}|\vec{\phi},z,\rho)]\} + constant \\ &\propto \exp\{\mathbf{E}_{q_z}[\ln p(\kappa_{i,k}|z_{u_1 i},\rho_{u_1 i},\ldots,z_{u_N i},\rho_{u_N,i})]\} \end{aligned}$$

Since Beta distribution is the conjugate distribution of the Binomial distribution, we have that $\kappa_{i,k}|z_{u_1 i},\rho_{u_1 i}\ldots z_{u_N i},\rho_{u_N i}$ follows a Beta distribution ($\kappa_{i,k}$ follows a Beta distribution and $\rho_{u,i}$ are Binomial distributions) with the following parameters:

$$\begin{aligned} \kappa_{i,k}|z_{u_1 i},\rho_{u_1 i},\ldots,z_{u_N i},\rho_{u_N i} &\sim Beta\left(\beta+\sum_{u,r_{u,i}\neq\bullet}\rho_{u,i}\cdot I(z_{u,i}=k),\right. \\ &\left.\beta+\sum_{u,r_{u,i}\neq\bullet}(R-\rho_{u,i})\cdot I(z_{u,i}=k)\right) \\ &= Beta\left(\beta+\sum_{u,r_{u,i}\neq\bullet}r^{+}_{u,i}\cdot I(z_{u,i}=k),\beta+\sum_{u,r_{u,i}\neq\bullet}r^{-}_{u,i}\cdot I(z_{u,i}=k)\right) \end{aligned}$$

Therefore,

$$\begin{aligned} q_\kappa(\kappa_{i,k}) &\propto \exp\{\mathbf{E}_{q_z}[\ln p(\kappa_{i,k}|z_{u_1,i},\rho_{u_1,i},\ldots,z_{u_N,i},\rho_{u_N,i})]\} \\ &= \exp\left\{\mathbf{E}_{q_z}\left[\ln\frac{\kappa_{i,k}^{\beta+\sum_{u,r_{u,i}\neq\bullet}r^{+}_{u,i}\cdot I(z_{u,i}=k)-1}(1-\kappa_{i,k})^{\beta+\sum_{u,r_{u,i}\neq\bullet}r^{-}_{u,i}\cdot I(z_{u,i}=k)-1}}{B(\beta,\beta)}\right]\right\} \\ &\propto \exp\left\{\mathbf{E}_{q_z}\left[\left(\beta+\sum_{u,r_{u,i}\neq\bullet}r^{+}_{u,i}\cdot I(z_{u,i}=k)-1\right)\ln\kappa_{i,k}\right.\right. \\ &\quad\left.\left.+\left(\beta+\sum_{u,r_{u,i}\neq\bullet}r^{-}_{u,i}\cdot I(z_{u,i}=k)-1\right)\ln(1-\kappa_{i,k})\right]\right\} \\ &= \exp\left\{\left(\beta+\sum_{u,r_{u,i}\neq\bullet}r^{+}_{u,i}\cdot\mathbf{E}_{q_z}[I(z_{u,i}=k)]-1\right)\ln\kappa_{i,k}\right. \\ &\quad\left.+\left(\beta+\sum_{u,r_{u,i}\neq\bullet}r^{-}_{u,i}\cdot\mathbf{E}_{q_z}[I(z_{u,i}=k)]-1\right)\ln(1-\kappa_{i,k})\right\} \\ &= \exp\left\{\left(\beta+\sum_{u,r_{u,i}\neq\bullet}r^{+}_{u,i}\cdot q_z(z_{u,i}=k)-1\right)\ln\kappa_{i,k}\right. \\ &\quad\left.+\left(\beta+\sum_{u,r_{u,i}\neq\bullet}r^{-}_{u,i}\cdot q_z(z_{u,i}=k)-1\right)\ln(1-\kappa_{i,k})\right\} \end{aligned}$$

According the following definitions:

- $\lambda_{u,i,k}=q_z(z_{u,i}=k)$ (we have defined it above).
- $\epsilon^{+}_{i,k}=\beta+\sum_{u,r_{u,i}\neq\bullet}r^{+}_{u,i}\lambda_{u,i,k}$
- $\epsilon^{-}_{i,k}=\beta+\sum_{u,r_{u,i}\neq\bullet}r^{-}_{u,i}\lambda_{u,i,k}$

we have that:

$$\begin{aligned} q_\kappa(\kappa_{i,k}) &\propto \exp\left\{\left(\beta+\sum_{u,r_{u,i}\neq\bullet}r^{+}_{u,i}\cdot\lambda_{u,i,k}-1\right)\ln\kappa_{i,k}\right. \\ &\quad\left.+\left(\beta+\sum_{u,r_{u,i}\neq\bullet}r^{-}_{u,i}\cdot\lambda_{u,i,k}-1\right)\ln(1-\kappa_{i,k})\right\} \\ &= \exp\left\{(\epsilon^{+}_{i,k}-1)\ln\kappa_{i,k}+(\epsilon^{-}_{i,k}-1)\ln(1-\kappa_{i,k})\right\} \\ &= (\kappa_{i,k})^{\epsilon^{+}_{i,k}-1}(1-\kappa_{i,k})^{\epsilon^{-}_{i,k}-1} \end{aligned}$$

Consequently, we have that $q_{\kappa_{i,k}}$, the approximated conditional probability distribution of $\kappa_{i,k}$ knowing the rating matrix, also follows a Beta distribution:

$$q_{\kappa_{i,k}} \sim Beta(\epsilon^{+}_{i,k},\epsilon^{-}_{i,k})$$

- Derivation of distribution (5) and Eq. (9). Since z is a categorical distribution, the conditional probability distribution of $z_{u,i}$ knowing the rating matrix also follows a categorical distribution.

$$\begin{aligned} \lambda_{u,i,k} = q_{z_{u,i}}(z_{u,i}=k) &\propto \exp\{\mathbf{E}_{q_{\vec{\phi}},q_\kappa}[\ln p(\vec{\phi},\kappa,z,\rho)]\}\propto \\ &\exp\{\mathbf{E}_{q_{\vec{\phi}},q_\kappa}[\ln(p(z_{u,i}=k|\vec{\phi}_u)\cdot p(\rho_{u,i}|\kappa_{i,k}))]\}\propto \\ &\exp\{\mathbf{E}_{q_{\vec{\phi}},q_\kappa}[\ln p(z_{u,i}=k|\vec{\phi}_u)+\ln p(\rho_{u,i}|\kappa_{i,k})]\}\propto \\ &\exp\{\mathbf{E}_{q_{\vec{\phi}}}[\ln p(z_{u,i}=k|\vec{\phi}_u)]+\mathbf{E}_{q_\kappa}[\ln p(\rho_{u,i}|\kappa_{i,k})]\}\propto \end{aligned}$$

Since $z_{u,i}$ is a categorical distribution with parameter $\vec{\phi}_u$ we have that

$$p(z_{u,i}|\vec{\phi}_u)=\phi_{u,k}$$

Since $\rho_{u,i}$ is a Binomial distribution with parameters $\kappa_{i,k}$ and R we have that

$$p(\rho_{u,i}|\kappa_{i,k}) \propto (\kappa_{i,k})^{\rho_{u,i}}(1-\kappa_{i,k})^{R-\rho_{u,i}}$$

Therefore, we have that:

$$\begin{aligned} \lambda_{u,i,k} &\propto \exp\{\mathbf{E}_{q_{\vec{\phi}}}[\ln\phi_{u,k}]+\mathbf{E}_{q_\kappa}[\ln p(\rho_{u,i}|\kappa_{i,k})]\} \\ &\propto \exp\{\mathbf{E}_{q_{\vec{\phi}}}[\ln\phi_{u,k}]+\rho_{u,i}\mathbf{E}_{q_\kappa}[\ln\kappa_{i,k}] \\ &\quad+(R-\rho_{u,i})\cdot\mathbf{E}_{q_\kappa}[\ln(1-\kappa_{i,k})]\} \\ &= \exp\{\mathbf{E}_{q_{\vec{\phi}}}[\ln\phi_{u,k}]+r^{+}_{u,i}\mathbf{E}_{q_\kappa}[\ln\kappa_{i,k}]+r^{-}_{u,i}\cdot\mathbf{E}_{q_\kappa}[\ln(1-\kappa_{i,k})]\} \end{aligned}$$

The following are properties of the Dirichlet and Beta distributions:

- $\mathbf{E}_{q_{\vec{\phi}}}[\ln\phi_{u,k}]=\Psi(\gamma_{u,k})-\Psi(\sum_{k=1}^{K}\gamma_{u,k})$
- $\mathbf{E}_{q_\kappa}[\ln\kappa_{i,k}]=\Psi(\epsilon^{+}_{i,k})-\Psi(\epsilon^{+}_{i,k}+\epsilon^{-}_{i,k})$
- $\mathbf{E}_{q_\kappa}[\ln(1-\kappa_{i,k})]=\Psi(\epsilon^{-}_{i,k})-\Psi(\epsilon^{+}_{i,k}+\epsilon^{-}_{i,k})$

where Ψ is the digamma function defined as the logarithmic derivative of the gamma function.
Therefore,

$$\begin{aligned} \lambda_{u,i,k} \propto \exp\big(&\Psi(\gamma_{u,k})+r^{+}_{u,i}\cdot\Psi(\epsilon^{+}_{i,k}) \\ &+r^{-}_{u,i}\cdot\Psi(\epsilon^{-}_{i,k})-R\cdot\Psi(\epsilon^{+}_{i,k}+\epsilon^{-}_{i,k})\big) \end{aligned}$$

Algorithm. The algorithm described in Section 4.5 is a coordinate-ascent algorithm, in which we iteratively optimize each variational parameter while holding the others fixed.

References

[1] Z. Ghahramani, M. Beal, Propagation algorithms for variational Bayesian learning, NIPS (2001) 507–513.
[2] D.M. Blei, A.Y. NG, M.I. Jordan, Latent Dirichlet Allocation, Advances in Neural Information Processing Systems, MIT Press, Cambridge, Mass, 2002.
[3] T. Hofmann, Probabilistic latent semantic indexing, Proceedings of the 22nd ACM-SIGIR, International Conference on Research and Development in Information Retrieval (Berkeley, Calif.), ACM, New York, 1999, pp. 50–57.
[4] S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, R.A. Harshman, Indexing by latent semantic analysis, J. ASIS 41 (6) (1990) 391–407.
[5] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, Adv. Neural Inf. Process. Syst. 20 (NIPS07) (2008) 1257–1264.
[6] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[7] C.M. Bishop, Pattern Recognition and Machine Learning, Information Science and Statistics. Springer, 2006.
[8] D. Lee, H. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.
[9] J. Bobadilla, F. Ortega, A. Hernando, J. Bobadilla, F. Ortega, A. Hernando, A collaborative filtering similarity measure based on singularities, Inf. Process. Manag. 48 (2) (2012) 204–217.
[10] J. Bobadilla, A. Hernando, F. Ortega, A. Gutirrez, Collaborative filtering based on significances, Inf. Sci. 185 (1) (2012) 1–17.
[11] J. Bobadilla, F. Serradilla, J. Bernal, A new collaborative filtering metric that improves the behavior of recommender systems, Knowl. Based Syst. 23 (2010) 520–528.
[12] J. Bobadilla, F. Ortega, A. Hernando, A. Arroyo, A balanced memory-based collaborative filtering similarity measure, Int. J. Intell. Syst. 27 (10) (2013) 939–946.

[13] C. Wang, D.M. Blei, Collaborative topic modeling for recommending scientific articles, in: Proceedings of the ACM SIGKDD, KDD '11, 2011, pp. 448–456.
[14] J. Kang, K. Lerman, LA-CTR: a limited attention collaborative topic regression for social media, in: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI), 2013.
[15] Y. Hu, Y. Koren, C. Volinky. Collaborative filtering for implicit feedback datasets, in: IEEE International Conference on Data Mining (ICDM) 2008.
[16] A. Hernando, J. Bobadilla, F. Ortega, J. Tejedor, Incorporating reliability measurements into the predictions of recommender system, Inf. Sci. 218 (2013) 1–16.
[17] A. Hernando, J. Bobadilla, F. Ortega, A. Gutirrez, Trees for explaining recommendations made through collaborative filtering, Inf. Sci. 218 (2013) 1–17.
[18] T. Hofmann, Latent semantic models for collaborative filtering, ACM Transactions on Information Systems (TOIS) 22 (1) (2004) 89–115.
[19] P. Gopalan, J.M. Hofman, D.M. Blei. Scalable Recommendations with Poisson Factorization, 2014, arXiv:1311.1704.
[20] C. Ding, T. Li, W. Peng, On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing, Computational Statistics Data Analysis 52 (2008) 3913–3927.
[21] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[22] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowl. Based Syst. 46 (2013) 109–132.
[23] S.K. Lee, Y.H. Cho, S.H. Kim, Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations, Inf. Sci. 180 (11) (2010) 2142–2155.
[24] A.B. Barragáns-Martínez, E. Costa-Montenegro, J.C. Burguillo, M. Rey-López, F.A. Mikic-Fonte, A. Peleteiro, A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition, Inf. Sci. 180 (22) (2010) 4290–4311.
[25] E.R. Núez-Valdéz, J.M. Cueva-Lovelle, O. Sanjuán-Martínez, V. García-Díaz, P. Ordoez, C.E. Montenegro-Marín, Implicit feedback techniques on recommender systems applied to electronic books, Comput. Human Behav. 28 (4) (2012) 1186–1193.
[26] J. Bobadilla, F. Serradilla, A. Hernando, Collaborative filtering adapted to recommender systems of e-learning, Knowl. Based Syst. 22 (2009) 261–265.
[27] J.J. Castro-Sanchez, R. Miguel, D. Vallejo, L.M. López-López, A highly adaptive recommender system based on fuzzy logic for b2c e-commerce portals, Expert Syst. Appl. 38 (3) (2011) 2441–2454.
[28] C. Ding, L. Tao, M.I. Jordan, Convex and semi-nonnegative matrix-factorizations, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 45–55.
[29] J. Wu, L. Chen, Y.-P. Feng, Z.-B. Zheng, M.C. Zhou, Z. Wu, Predicting quality of service for selection by neighborhood-based collaborative-filtering, IEEE Trans. Syst. Man Cybern. Syst. 43 (2013) 428–439.
[30] N.-Y. Guan, D.-C. Tao, Z.-G. Luo, B. Yuan, Online nonnegative matrix factorization with robust stochastic approximation, IEEE Trans. Neural Netw. Learn. Syst. 23 (2012) 1087–1099.
[31] X. Luo, M.-C. Zhou, Y.-N. Xia, Q.-S. Zhu, An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems, IEEE Trans. Ind. Inf. 10 (2014) 1273–1284.
[32] Q. Wu, M. Tan, X. Li, H. Min, N. Sun, NMFE-SSCC: Non-negative matrix factorization ensemble for semi-supervised collective classification Knowl, Based Syst. 89 (2015) 160–172.
[33] Incorporating group recommendations to recommender systems: alternatives and performance, Inf. Process. Manag. 49 (2013) 895–901.