Data Mining and Knowledge Discovery manuscript No.
(will be inserted by the editor)

An Optimization Model for Collaborative Recommendation Using a Covariance-Based Regularizer

Fabian Lecron · François Fouss

Received: date / Accepted: date

Abstract This paper suggests a convex regularized optimization model to produce recommendations, which is adaptable, fast, and scalable - while remaining very competitive to state-of-the-art methods in terms of accuracy. We introduce a regularizer based on the covariance matrix such that the model minimizes two measures ensuring that the recommendations provided to a user are guided by both the preferences of the other users in the system and the known preferences of the user being processed. It is adaptable since (1) it can be viewed from both user and item perspectives (allowing to choose, depending on the task, the formulation with fewer decision variables) and (2) multiple constraints depending on the context (and not only based on the accuracy, but also on the utility of personalized recommendations) can easily be added, as shown in this paper through two examples. Since our regularizer is based on the covariance matrix, this paper also describes how to improve computational and space complexities by using matrix factorization techniques in the optimization model, leading to a fast and scalable model. To illustrate all these concepts, experiments were conducted on four real datasets of different sizes (i.e., FilmTrust, Ciao, MovieLens, and Netflix) and comparisons with state-of-the-art methods are provided, showing that our context-sensitive approach is very competitive in terms of accuracy.

This is a post-peer-review, pre-copyedit version of an article published in Data Mining & Knowledge Discovery. The final authenticated version is available online at: https://doi.org/10.1007/s10618-018-0552-3

F. Lecron
Department of Engineering Innovation Management
Faculty of Engineering, University of Mons, Belgium
E-mail: fabian.lecron@umons.ac.be

F. Fouss
Louvain School of Management (LSM)
Louvain Research Institute in Management and Organizations (LouRIM)
Université catholique de Louvain (UCL), Belgium
E-mail: francois.fouss@uclouvain.be

Keywords Recommender system · Collaborative filtering · Regularizer · Convex optimization · Matrix factorization

1 Introduction

1.1 General Introduction

Recommending items to family members, friends, or neighbors is a social process which is part of human life since the dawn of time: before going to the cinema, before buying a car, before booking holidays, etc., we behave in the same way, collecting the opinions of relatives and experts before making a decision. Each of us is daily confronted with recommendations, as were our ancestors and as will be our children.

The process of recommendation has however drastically evolved last decades since part of our lives now takes place when connected. But, as the amount of available information has been growing at a phenomenal rate, it is more and more difficult to process it. Everybody has already been overwhelmed with the number of new books, journal articles, and conference proceedings coming out each year. During the last decades, technology has dramatically made publishing and distributing information easier. The challenge now consists in developing technologies that can help us sift through all this available information to find the most valuable information for each of us. Recommender systems are one of these technologies. Recommender systems try to provide people with recommendations of items they will appreciate, based on their past preferences, history of purchase, and demographic information.

Recommender systems have their origin (see the surveys of Adomavicius and Tuzhilin (2005), or Lü et al (2012) for more details) in the work done in, mainly, information retrieval (Salton, 1989), cognitive science (Rich, 1979), forecasting theories (Armstrong, 2001), marketing (Lilien et al, 1992), management (Murthi and Sarkar, 2003), and emerged as an independent research area in the mid-1990s, with the first papers on collaborative filtering appearing in (Hill et al, 1995; Resnick et al, 1994; Shardanand and Maes, 1995). Various approaches (content-based, collaborative, or hybrid approaches – combining the first two) were developed through years to produce the *best* recommender systems (see, e.g., (Adomavicius and Tuzhilin, 2005; Lü et al, 2012)). While content-based approaches recommend items similar to the ones a user preferred in the past depending on the features of the items, collaborative approaches recommend to a user items that people with similar tastes and preferences have liked or items similar to the ones the considered user has preferred, depending, this time, on the links between items and users, and not on the features of items.

This work suggests to produce recommendations by using a convex regularized optimization model (i.e., a linear least-squares), minimizing two measures. Moreover, we introduce a regularizer based on the covariance matrix ensuring that the recommendations provided to a user are guided by both the prefer-

ences of the other users in the system (our model can therefore be classified as a collaborative filtering one – see Related Work) and the known preferences of the user being processed, and leads to very competitive results in terms of accuracy. This is very attractive since it is now clear in the recommender systems' community that state-of-the-art systems, based on various approaches, can provide very accurate recommendations for different kinds of items in many areas. It is however less evident that only the accuracy still has to be taken into account for designing and comparing recommender systems. Imagine you need some novelty in your recommendations, and therefore a system that favors non-popular items, or imagine you need to add a financial constraint in the model (e.g., on your margin), etc. Such constraints should be integrated into a recommender system. Interestingly, multiple constraints, depending on the context, can easily be added to an optimization model such as ours, as is also shown in this paper.

1.2 Related Work

The two primary areas of collaborative filtering are distinguished according to the neighborhood methods and the latent factor models (Koren, 2008; Koren et al, 2009). Approaches based on *neighborhoods* use various statistical techniques to determine a set of users/items, often referred to as the neighbors, who are the most similar to the active user/to the items the considered user has preferred, depending on the historical behavior of the users. The most popular approaches (Herlocker et al, 2002) use the Pearson correlation coefficient, the Spearman rank correlation coefficient, and the cosine correlation as measure of similarity. Many extensions to these classical techniques as well as other neighborhoods measures were proposed in the literature. Approaches based on *latent factor models* consist in transforming both items and users to the same latent factor space, making them directly comparable (Koren, 2008). Latent factor models can be obtained by performing matrix factorization. In a basic form, the factor vectors are deduced with a minimization of the regularized squared error on a set of known ratings. Singular Value Decomposition (SVD) is a well-established technique to obtain latent factors in information retrieval (Koren et al, 2009; Paterek, 2007; Koren, 2009). Other matrix factorization methods have been used in the literature: principal component analysis (Kim and Yum, 2005; Yu et al, 2009), bounded matrix factorization (Kannan et al, 2012), nonnegative matrix factorization (Zhang et al, 2006; Luo et al, 2014), probabilistic matrix factorization (Salakhutdinov and Mnih, 2008; Shan and Banerjee, 2010), etc.

One of the first approach to merge latent factor and neighborhoods models is described in (Koren, 2008). The authors added a term allowing to exploit implicit feedback when factorizing the matrix of known ratings. Following the same idea, Zhang et al (2013) integrated a social regularizer and an item similarity regularizer into a probabilistic matrix factorization. The advantage was to use the social interactions of users on a microblog. A graph-regularized

nonnegative matrix factorization was proposed by Gu et al (2010) to include user's demographic information, social interactions, item's genre information, etc. The additional term is defined with a graph Laplacian. Recently, Rao et al (2015) proposed a scalable algorithm for matrix completion that incorporates additional structural information (e.g., social graphs, item co-purchasing graphs). The proposed regularizer relies on a Laplacian matrix associated to the graph encoding relationships between variables.

In the present paper, we introduce a regularizer based on the inverse covariance matrix. The interest of using the inverse covariance matrix is that it encodes the correlations between all items or all users (we will see that we can do both). Some similar regularizers using covariance matrix through a trace norm are proposed in (Feldman, 2012) in the context of multi-task learning (MTL) but the purpose is quite different with the present work. Still in (Feldman, 2012), other regularizers usually applied for multi-task learning are presented: a distance to mean regularizer, a trace norm regularizer and a pairwise distance regularizer. In our opinion, using correlations thanks to the inverse covariance matrix provides a better generalization than using a simple distance to mean or a pairwise distance regularizer. In a similar way, Ning and Karypis (2011) also proposed a regularized optimization problem based on a $n \times n$ (n being the number of items) aggregation matrix. In order to avoid a dimensionality problem, they introduce the l_1-norm of the aggregation matrix to learn a sparse matrix. Theoretically, we could merge our regularizer into a matrix factorization technique such as suggested by the methods described above (Gu et al, 2010; Zhang et al, 2013; Rao et al, 2015). However, these methods work well with sparse similarity matrices and using our covariance-based regularizer would have been too computationally demanding. To give an order of magnitude, a user-user similarity matrix for the Netflix dataset (Bennett and Lanning, 2007) would require to deal with $480,189 \times 480,189$ links between users.

For these reasons, matrix factorization is included in our model as a preprocessing step and its role is somehow different from latent factor models. In our approach, it only serves to express users or items in the same space and to reduce the dimensionality of the problem. The results of matrix factorization is then used in the convex problem optimizing the similarity measure we have defined. As a consequence, our model could be classified as a neighborhood one since we use a similarity measure between users/items (two formulations are defined) written as a convex optimization problem.

The positive impact of performing a matrix factorization and then writing our recommender system as a convex optimization model is that we can easily include additional constraints in the problem depending on the context. In their well-known survey, Adomavicius and Tuzhilin (2005) highlighted the fact that including multiple criteria in recommender systems to extend their capabilities was a challenging issue. Since then, some works have focused on this topic.

Rodriguez et al (2012) studied a specific application of recommender systems with multiple objectives. They described *TalentMatch*, a product at

LinkedIn which scours the database of users and finds the best candidates for a given job posted on the website. The system provides recommendations by computing similarities between a subset of a member's feature vector and semantically related subsets of the given job's feature vector. The authors argued in the paper that it is interesting to add other features than semantic ones in the *TalentMatch* system, such as the job-seeking intent of a member. By doing so, they showed that the semantic match is decreased but that the utility of the recommendations is increased (42% increase on email reply rate). The specificity of their approach is that they use additional objectives in an optimization problem but they also control the loss of matching precision by penalizing the distance between the semantic match distribution (without additional constraints) and the enhanced score distribution (with additional constraints). The optimization problem is different from what we propose in this paper since objective function and constraints in (Rodriguez et al, 2012) can be nonlinear. It is unfortunately impossible to estimate the influence on optimization time since this aspect is not discussed in their paper.

The advantage of focusing on the utility of personalized recommendations (and not only on the accuracy) is also stressed in (Jambor and Wang, 2010). The authors experimented a recommender system based on the formulation of a problem using constrained linear optimization techniques. The linear constraints allow to solve the problem faster than in (Rodriguez et al, 2012). The objective function of the problem is very simple and linked to constraints such as the fact that a user likes popular items or the probability that items are available during recommendation. The authors pointed out that the accuracy is drastically reduced with their approach. Therefore, they propose to enrich the objective function with a nonlinear term based on covariance matrix without explaining how they deal with the dimensionality of the problem.

Agarwal et al (2011) proposed several formulations of constrained linear problems with multiple objectives applied to a Yahoo module providing users with interesting and informative articles everyday. The idea was to optimize jointly for number of clicks and other post-clicks utilities such as total time spent. The users are segmented into homogeneous groups, and features such as number of clicks and post-clicks utilities are computed for each segment. Their work is therefore different to the approach detailed in the present paper since we are interested in collaborative filtering for personalized recommendations.

A common issue with the aforementioned methods is that they all are specific to the users. In the literature, some datasets are treated effectively by user-based methods and others by item-based methods. As a consequence, we propose a general framework that can be formulated either in a user-based form or in an item-based one.

1.3 Contributions and Organization of the Paper

This work has 4 main contributions:

- It defines a new framework for recommending items, exploiting a convex optimization model using a similarity measure based on the covariance matrix.
- It shows how the optimization of this problem, although already fast, can be made even faster and scalable by using matrix factorization, making the model suited for many applications where real-time recommendations are required for consumers.
- It shows how multiple constraints depending on the context can be easily added to such an optimization model, providing new opportunities for more diversified and adapted recommender systems.
- It shows that both user-based and item-based approaches can be integrated in this framework, allowing to choose, in a very attractive way and depending on the task, the formulation with fewer decision variables.

Section 2 describes the convex optimization framework for collaborative recommendation and further investigates the dimensionality reduction work. In Section 3, the optimization model is applied on well-known datasets in the field of recommender systems, and results are shown and analyzed. Concluding remarks and possible extensions are discussed in Section 4.

2 Methods

2.1 Framework Overview

The goal of a recommender system is to provide an ordered list of items to each user. The items are ordered given a measure of preference that the user would give to them. In the present work, we propose to predict these preferences by solving an optimization problem. Regardless of the context, we distinguish two measures that are required to be optimized to obtain relevant recommendations, and we introduce a regularizer based on the covariance matrix.

In the following, we consider that the number of users is equal to u and the number of items is equal to v.

2.1.1 Regularizer Based on Covariance Matrix

In this section, we present a user-oriented point of view of the problem. Section 2.2.2 shortly describes the dual formulation of the problem with an item-oriented point of view.

Let us define the decision variable $\mathbf{X} \in \mathbb{R}^{v \times u}$, a matrix such as each column represents a user and each row an item:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_j, \ldots, \mathbf{x}_u]\,. \tag{1}$$

Note that $\mathbf{x}_j$ is a column vector characterizing a user j such as: $\mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{ji}, \ldots, x_{jv})^{\mathbf{T}}$.

First, the known preferences of the user being processed by the system need to guide the process. To illustrate our purpose, let us consider a matrix

$\mathbf{M}$ where each column stands for a user and each row for an item, in such a way that m_{ij} represents the preference (e.g., an implicit feedback with a value of 1 or 0) of a user j for an item i. If the user did not expressed a preference then m_{ij} is set to 0. Notice that, if the matrix is a rating matrix, a conflict could arise if the value 0 is also the lower bound of the rating scale. In this case, authors in (Devooght et al, 2015) showed the interest of using an explicit prior on unknown ratings. By assuming that unknown items are more likely to be weakly rated, they improved results over state-of-the-art matrix factorization methods. As a consequence, in order to avoid a conflict, we could use 0 as an explicit prior on unknown ratings in the matrix. Let $\mathbf{x}$ be the solution of the optimization model, representing the predicted preferences for a user j and all the items, let us note κ_j the set of items i for which m_{ij} is known at column j. Therefore, for each user j the difference between the known m_{ij} and the x_{ij} associated to these known m_{ij} has to be minimized. A vector is therefore built, for each user j, containing these differences $\forall i \in \kappa_j$:

$$\begin{pmatrix} \vdots \\ x_{ij} - m_{ij} \\ \vdots \end{pmatrix} \tag{2}$$

Actually, Equation (2) tries to get $\mathbf{x}_j$ close to the *a priori* information we know about the user being processed.

Secondly, we need to ensure that the recommendations provided to a user are guided by the preferences of the other users in the system. One way to take into account the preferences of all the users of the system is to minimize the Mahalanobis distance d between $\mathbf{x}_j$ and the sample characterized by $\mathbf{M}$, summarized by its mean and its covariance matrix. Formally, we have:

$$d = \sqrt{(\mathbf{x}_j - \bar{\mathbf{x}})^{\mathbf{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_j - \bar{\mathbf{x}})}, \tag{3}$$

where $\bar{\mathbf{x}}$ is the mean computed over the columns of $\mathbf{M}$ and $\boldsymbol{\Sigma}^{-1}$ is the inverse covariance matrix associated to $\mathbf{M}$. If $\mathbf{m}_j$ is the column vector j of matrix $\mathbf{M}$, we have that $\boldsymbol{\Sigma} = \frac{1}{u-1} \sum_{j=1}^{u} (\mathbf{m}_j - \bar{\mathbf{x}})(\mathbf{m}_j - \bar{\mathbf{x}})^{\mathbf{T}}$, where $\bar{\mathbf{x}} = \frac{1}{u} \sum_{j=1}^{u} \mathbf{m}_j$.

The Mahalanobis distance allows to include the preferences of all the users in the optimization process. It actually tries to get $\mathbf{x}_j$ close to the mean while taking into account the correlations between items (represented by $\boldsymbol{\Sigma}^{-1}$). These correlations are computed with the preferences (or implicit feedback) given by all the users to the items. Other regularizations are also possible. We have tested a Laplacian regularization (Smola and Kondor, 2003) without obtaining results as good as those obtained with the proposed regularization. It is important to note that in Equation (3), matrix $\boldsymbol{\Sigma}$ might not be invertible. However, Section 2.3 explains how to reduce the dimensionality of the problem and shows that we do not need to invert $\boldsymbol{\Sigma}$ but only a reduced version of it.

Finally, the two measures described in this section are used to form the optimization model. Moreover, effective recommender systems are required to be fast in order to deal with large amount data. Therefore, our system has

been designed by using convex optimization since convex problems can be solved very reliably and efficiently with a polynomial-time complexity (Boyd and Vandenberghe, 2004). The next section explains how to integrate our recommender system into a convex optimization framework.

2.2 Regularized Optimization Problem for Collaborative Recommendation

This section shows how to transform the two measures defined in Section 2.1.1 using a norm, allowing us to rewrite our problem as a convex optimization model (i.e., a linear least-squares). In the following, we distinguish the user-based case, the item-based case, and the case where neighbors are considered. The formulations of the recommender system proposed in this section are based on solving an unconstrained linear least-squares. In Matlab, the operation $\mathbf{A}\backslash\mathbf{B}$ returns a least-squares solution to the system of equations $\mathbf{Ax} = \mathbf{B}$.

2.2.1 User-based

A linear least-squares (Gill et al, 1981) is formulated according to:

$$\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{Ax} - \mathbf{b}\|_2^2 . \tag{4}$$

Note that the $\mathbf{x}$ solution of such a linear least-squares problem can be reduced to solving a set of linear equations $\left(\mathbf{A^T A}\right)\mathbf{x} = \mathbf{A^T b}$ (see for example (Boyd and Vandenberghe, 2004)). So, the analytical solution is given by $\mathbf{x} = \left(\mathbf{A^T A}\right)^{-1}\mathbf{A^T b}$.

To be in line with Equation (4), the regularizer based on Mahalanobis distance (see Equation (3)) needs to be transformed given:

$$\begin{aligned} d^2 &= (\mathbf{x}_j - \bar{\mathbf{x}})^{\mathbf{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_j - \bar{\mathbf{x}}) \\ &= (\mathbf{L^T}(\mathbf{x}_j - \bar{\mathbf{x}}))^{\mathbf{T}} \mathbf{L^T}(\mathbf{x}_j - \bar{\mathbf{x}}) \\ &= \left\|\mathbf{L^T}(\mathbf{x}_j - \bar{\mathbf{x}})\right\|_2^2 \\ &= \left\|\mathbf{L^T x}_j - \mathbf{L^T}\bar{\mathbf{x}}\right\|_2^2 , \end{aligned} \tag{5}$$

where $\mathbf{L}$ is the Cholesky decomposition of the inverse covariance matrix such as: $\boldsymbol{\Sigma}^{-1} = \mathbf{LL^T}$.

Moreover, Equation (2) is simply modified by considering the norm such as:

$$\sum_{i \in \kappa_j} (x_{ij} - m_{ij})^2 . \tag{6}$$

Equations (5) and (6) can then be associated in the convex model defined in Equation (4). These constraints actually need to be minimized, leading to the following formulation of the objective function:

$$\min_{\mathbf{x}_j} \frac{1}{2} \left\| \mathbf{L}^{\mathbf{T}} \mathbf{x}_j - \mathbf{L}^{\mathbf{T}} \bar{\mathbf{x}} \right\|_2^2 + \frac{1}{2} \sum_{i \in \kappa_j} (x_{ij} - m_{ij})^2. \tag{7}$$

In Equation (7), a parameter α could be added to adjust the impact of the regularization. However, we noticed in our experiments that when working with implicit feedback, a value of α near to 1 is a good trade-off. Let us assume $\mathbf{x}_j^\star$, the optimal solution associated to the minimization problem described in (7). It represents the predicted preferences of user j for all the items. Therefore, a number of recommendations can be made to this user by considering the items (not already purchased by the user) with the highest scores of preference.

Our approach uses the Mahalanobis distance to take into account the correlations between items and between users. Actually, the Mahalanobis distance tries to measure the difference between a vector $\mathbf{x}_j$ and the mean $\bar{\mathbf{x}}$ of the whole training set. Instead of using $\bar{\mathbf{x}}$, we could consider a set of nearest neighbors of the current decision variable. This way, we introduce an adapted Mahalanobis distance which represents the difference between $\mathbf{x}_j$ and the mean $\bar{\mathbf{x}}_{\text{neigh}}$ of the neighbors which is influenced by the number of neighbors (users in the user-based point of view) considered (this parameter needs to be fixed).

Several methods exist to compute a similarity measure between users or items in order to define neighborhoods (e.g., Pearson correlation coefficient in (Herlocker et al, 2002), kernels in (Fouss et al, 2012), etc. – see (Deza and Deza, 2014) or (Fouss et al, 2016) for a survey). We decided to use a simple (and known to provide good results) similarity measure based on the cosine correlation.

2.2.2 Item-based

The item-based formulation is proposed as a dual formulation of the user-based one. The decision variable is a matrix $\mathbf{X} \in \mathbb{R}^{u \times v}$ such as each column represents an item and each row a user:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_i, \ldots, \mathbf{x}_v]\,. \tag{8}$$

Note that $\mathbf{x}_i$ is a column vector characterizing an item i such as: $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{iu})^{\mathbf{T}}$.

Equation (7) is easily adapted by considering $\mathbf{M}^{\mathbf{T}}$ the transpose of matrix $\mathbf{M}$, where each column stands for an item and each row for a user, in such a way that m_{ji} represents the preference of a user j for an item i. Solving the item-based version of (7) provides an optimal solution $\mathbf{x}_i^\star$. It represents the predicted preferences of all the users for the item i. As a consequence, we have to compute the value of $\mathbf{x}_i^\star$ for $i \in (1, v)$. Finally, for each user, we recommend a given number of items with the highest preferences.

2.3 Dimensionality Reduction

The number of variables involved in Equation (7) can be important, especially if a large amount of items and users is considered. Even if convex optimization is characterized by polynomial-time complexity, it is relevant to reduce the number of variables in these equations. Furthermore, computing matrix $\mathbf{L}$ in this context is memory-demanding for large datasets since it requires to have access to the entire covariance matrix which is non-sparse. Matrix $\mathbf{L}$ should therefore be computed when the dimensionality of the problem has been reduced.

Two methods for reducing the dimensionality are considered in the present work: Principal Component Analysis and Nonnegative Matrix Factorization. Note that other matrix factorization techniques could have been used to reduce the dimensionality (e.g., convex formulations such as (Hsieh and Olsen, 2014)). The following section presents our framework with matrix factorization in a general way.

2.4 Matrix Factorization

Matrix factorization allows to decompose a matrix as the product of matrices. Let us consider $\mathbf{M}$, a matrix where each column stands for a user and each row for an item. By considering that $\mathbf{M}$ contains v rows and u columns, we can decompose $\mathbf{M}$ such as:

$$\mathbf{M} \approx \mathbf{\Phi}\mathbf{B}, \tag{9}$$

where $\mathbf{M}$ is a matrix where each column is a data point (i.e., a user), $\mathbf{B}$ is a $(r \times u)$ matrix containing the coordinates of these data points in the basis $\mathbf{\Phi}$, which is a $(v \times r)$ matrix, and r is the number of components, which lower than u or v (depending on the approach).

When considering the user-based case, a column vector $\mathbf{x}_j$ (a vector of preferences) of matrix $\mathbf{M}$ is given by:

$$\mathbf{x}_j = \mathbf{\Phi}\widehat{\mathbf{b}}_j, \tag{10}$$

where each column of $\mathbf{\Phi}$ is a basis element, and $\widehat{\mathbf{b}}_j$ is a vector representing the preferences of user j in the basis.

By reducing the dimensionality with matrix factorization, the second term of Equation (7) is written:

$$\frac{1}{2}\sum_{i \in \kappa_j} (\boldsymbol{\phi}_i \widehat{\mathbf{b}}_j - m_{ij})^2. \tag{11}$$

where $\boldsymbol{\phi}_i$ is the row of $\mathbf{\Phi}$ related to item i.

The first term of Equation (7) becomes:

$$\begin{aligned}\frac{1}{2}\left\|\mathbf{L^T}\mathbf{x}_j - \mathbf{L^T}\bar{\mathbf{x}}\right\|_2^2 &= \frac{1}{2}\left\|\mathbf{L^T}\left(\mathbf{\Phi}\widehat{\mathbf{b}}_j\right) - \mathbf{L^T}\bar{\mathbf{x}}\right\|_2^2 \\ &= \frac{1}{2}\left\|\mathbf{L^T}\left(\mathbf{\Phi}\widehat{\mathbf{b}}_j - \bar{\mathbf{x}}\right)\right\|_2^2 \\ &= \frac{1}{2}\left\|\mathbf{L_\Phi^T}\left(\widehat{\mathbf{b}}_j - \bar{\mathbf{x}}_\mathbf{\Phi}\right)\right\|_2^2,\end{aligned} \tag{12}$$

where $\mathbf{L_\Phi^T}$ and $\bar{\mathbf{x}}_\mathbf{\Phi}$ are the expressions of $\mathbf{L^T}$ and $\bar{\mathbf{x}}$ in the basis $\mathbf{\Phi}$. These expressions are deduced given that matrix decomposition allows to write: $\mathbf{L^T} = \mathbf{\Phi}\mathbf{L_\Phi^T}$ and $\bar{\mathbf{x}} = \mathbf{\Phi}\bar{\mathbf{x}}_\mathbf{\Phi}$. In order to compute $\mathbf{L_\Phi^T}$, the covariance matrix $\mathbf{\Sigma_B}$ related to $\mathbf{B}$ is computed, where $\mathbf{B}$ is the expression of $\mathbf{M}$ in the basis $\mathbf{\Phi}$ (we have indeed $\mathbf{M} = \mathbf{\Phi}\mathbf{B}$). If $\mathbf{b}_j$ is the column vector j of matrix $\mathbf{B}$, we have that $\mathbf{\Sigma_B} = \frac{1}{u-1}\sum_{j=1}^{u}(\mathbf{b}_j - \bar{\mathbf{x}}_\mathbf{\Phi})(\mathbf{b}_j - \bar{\mathbf{x}}_\mathbf{\Phi})^\mathbf{T}$. If r is the number of components, this covariance matrix $\mathbf{\Sigma_B}$ is a $r \times r$ matrix. Then, $\mathbf{L_\Phi^T}$ is obtained by performing a Cholesky decomposition of the inverse covariance matrix (by a simple call to the function *chol* on Matlab).

Finally, the user-based formulation with a reduced number of variables is given by:

$$\min_{\widehat{\mathbf{b}}_j} \frac{1}{2}\left\|\mathbf{L_\Phi^T}\left(\widehat{\mathbf{b}}_j - \bar{\mathbf{x}}_\mathbf{\Phi}\right)\right\|_2^2 + \frac{1}{2}\sum_{i\in\kappa_j}(\phi_i\widehat{\mathbf{b}}_j - m_{ij})^2 \tag{13}$$

When the nearest neighbors are included in the optimization problem, the first term of Equation (13) becomes:

$$\frac{1}{2}\left\|\mathbf{L_\Phi^T}\left(\widehat{\mathbf{b}}_j - \bar{\mathbf{x}}_{\text{neigh},\Phi}\right)\right\|_2^2, \tag{14}$$

where $\mathbf{L_\Phi^T}$, $\bar{\mathbf{x}}_\mathbf{\Phi}$, and $\bar{\mathbf{x}}_{\text{neigh.},\Phi}$ are the expression of, respectively, $\mathbf{L^T}$, $\bar{\mathbf{x}}$, and $\bar{\mathbf{x}}_{\text{neigh}}$ in the basis $\mathbf{\Phi}$. The item-based formulation with a reduced number of variables is obtained with a similar approach.

Solving the problem defined in Equation (13) provides the optimal vector $\widehat{\mathbf{b}}^\star$. This vector needs to be mapped in the initial space by knowing that $\mathbf{x}^\star = \mathbf{\Phi}\widehat{\mathbf{b}}^\star$, and the resulting vector $\mathbf{x}^\star$ represents the predicted preferences of a user for all the items. The items with the highest preference are finally recommended to the user. Let us note that this process needs to be separately repeated for all the users. In the item-based formulation, the optimal vector $\widehat{\mathbf{b}}^\star$ allows to compute the preferences of all users for one item.

2.4.1 Nonnegative Matrix Factorization

Nonnegative Matrix Factorization (NMF) is a linear dimensionality reduction technique for nonnegative data. NMF has become a widely used tool for the analysis of high-dimensional data as it automatically extracts sparse and meaningful features from a set of nonnegative data vectors (Gillis, 2014). The technique is relevant in our context since the ratings or implicit feedback, and as a consequence the preferences of users, are always nonnegative.

In order to factorize $\mathbf{M}$, the problem to be solved is written:

$$\min_{\mathbf{\Phi},\mathbf{B}} \|\mathbf{M} - \mathbf{\Phi}\mathbf{B}\|_F^2 \quad \text{with } \mathbf{\Phi} \geq 0 \text{ and } \mathbf{B} \geq 0. \tag{15}$$

To solve this problem, we use the accelerated hierarchical alternating least squares (HALS) algorithm described in Section 4.2 of (Gillis, 2011). The source code is available online[1]. Solving this problem provides the matrices $\mathbf{\Phi}$ and $\mathbf{B}$ which are required to obtain $\mathbf{L}_{\mathbf{\Phi}}^{\mathbf{T}}$ and $\bar{\mathbf{x}}_{\mathbf{\Phi}}$ in Equation (13).

2.4.2 Principal Component Analysis

Principal Component Analysis (PCA) is a well-known statistical method to reduce the number of variables (Johnson and Wichern, 2002). PCA decomposes the original space into orthogonal axes along which the variance is maximized.

In the case of PCA, the vector $\mathbf{x}_j$ is not represented by Equation (10) but by:

$$\mathbf{x}_j = \bar{\mathbf{x}} + \mathbf{\Phi}\widehat{\mathbf{b}}_j, \tag{16}$$

where $\bar{\mathbf{x}}$ is a column vector characterizing the mean of users computed over matrix $\mathbf{M}$, $\mathbf{\Phi}$ is a matrix containing the orthogonal axes computed from matrix $\mathbf{M}$, and $\widehat{\mathbf{b}}_j$ is a column vector containing the weights on these axes related to the user j.

Given Equation (16), Equation (12) needs to be slightly adapted by replacing $\mathbf{x}_j$ with $\bar{\mathbf{x}} + \mathbf{\Phi}\widehat{\mathbf{b}}_j$. Then, the first term of Equation (13) becomes:

$$\begin{aligned}
\frac{1}{2}\left\|\mathbf{L}^{\mathbf{T}}\mathbf{x}_j - \mathbf{L}^{\mathbf{T}}\bar{\mathbf{x}}\right\|_2^2 &= \frac{1}{2}\left\|\mathbf{L}^{\mathbf{T}}\left(\bar{\mathbf{x}} + \mathbf{\Phi}\widehat{\mathbf{b}}_j\right) - \mathbf{L}^{\mathbf{T}}\bar{\mathbf{x}}\right\|_2^2 \\
&= \frac{1}{2}\left\|\mathbf{L}^{\mathbf{T}}\mathbf{\Phi}\widehat{\mathbf{b}}_j\right\|_2^2 \\
&= \frac{1}{2}\left\|\mathbf{Diag}\left(\frac{1}{\boldsymbol{\sigma}}\right)\widehat{\mathbf{b}}_j\right\|_2^2.
\end{aligned} \tag{17}$$

Since the components of $\mathbf{B}$ are uncorrelated in PCA, only the diagonal of $\mathbf{L}^{\mathbf{T}}$ remains, which is represented by the inverse of the variance on each component. In this specific case, $\mathbf{L}^{\mathbf{T}}$ does not have to be formally computed, only $\boldsymbol{\sigma}^2$, a vector where each element is the variance of one component of PCA, is needed.

In order to obtain matrices $\mathbf{\Phi}$ and $\mathbf{B}$, a Singular Value Decomposition (SVD) is performed on the centered version $\tilde{\mathbf{M}}$ of $\mathbf{M}$. In the literature, SVD on a given $(m \times n)$ matrix $\mathbf{M}$ is often characterized by: $\mathbf{M} = \mathbf{USV}^{\mathbf{T}}$. If we only consider the r largest singular values (e.g., with function *svds* on Matlab), $\mathbf{\Phi}$ is equal to $\mathbf{U}$ while $\mathbf{B}$ is equal to $\mathbf{SV}^{\mathbf{T}}$ when SVD is performed on $\tilde{\mathbf{M}}$.

[1] `https://sites.google.com/site/nicolasgillis/code`

3 Experiments

3.1 Data and Methods

In order to assess the effectiveness of our approach, we used four datasets of different sizes. The first one is FilmTrust, a small dataset crawled from the FilmTrust website (Guo et al, 2013). It contains 35,497 ratings (from 1 to 5) given by 1,508 users to 2,071 movies. Tang et al (2012) crawled the popular product review website `www.ciao.co.uk`. We used these data to create our second dataset with 102,616 ratings given by 7,010 users to 20,000 items (here, the number of items is greater than the number of users). The third dataset is the MovieLens 10M dataset[2]. It consists of 10,000,054 ratings (from 1 to 5) given by 71,567 users to 10,681 movies (here, the number of users is greater than the number of items). It is considered that each user has rated at least 20 movies. Finally, we made experiments with the Netflix dataset (Bennett and Lanning, 2007), containing 100,480,507 ratings (from 1 to 5) given from 480,189 users to 17,770 movies.

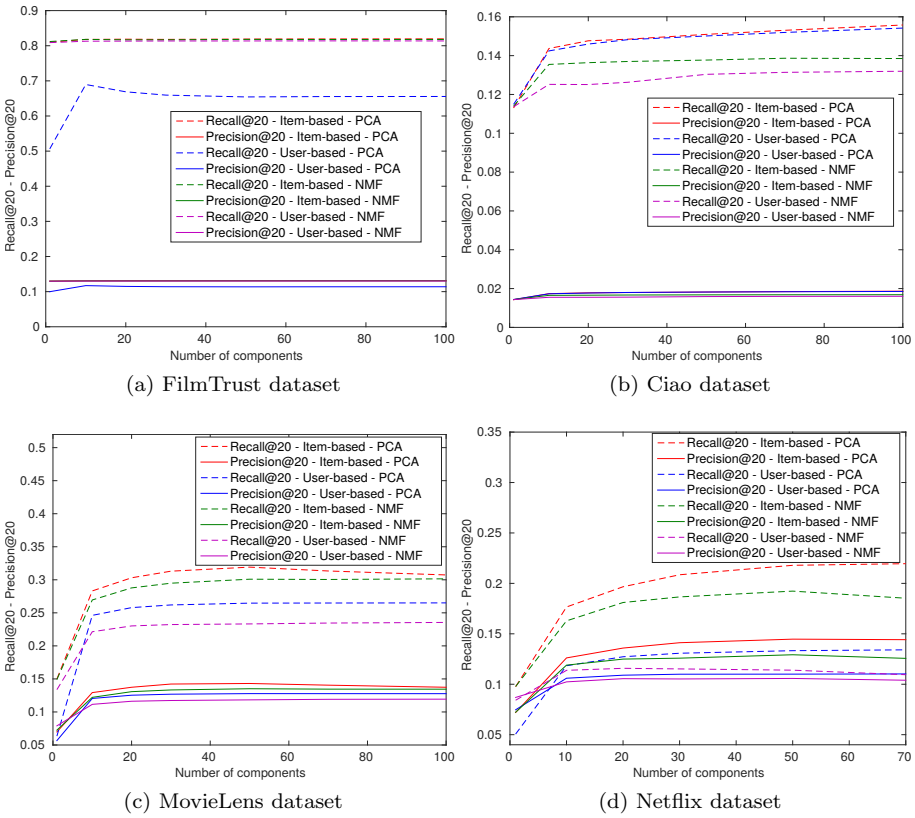
For the four datasets, we worked with implicit feedback given by the users since our recommender system is not designed to predict ratings but to provide a ranking given some predicted preferences. It is therefore relevant not to consider the actual value of the ratings but to consider a 1 when a user has rated an item and a 0 otherwise.

In the following experiments, the datasets were randomly divided into n subsets (with $n = 10$) and the algorithms were executed n times (n-fold cross-validation). In each run, one of the n subsets (containing about 10% of the ratings) was used as the test set while the other $n - 1$ subsets were merged into a training set. Then, the average result was computed on the n runs.

For each user and each run, the algorithm computes, based on the training set, a ranked list of preferences about items. From that information, we retain a ranked list of all the items that the user has not rated, according to the training set.

Regarding the evaluation of a recommender system, the literature is often divided between accuracy metrics (precision, recall, etc.) and error metrics (RMSE, MAE, etc.). Cremonesi et al (2010) suggested that there is no monotonic relation between accuracy metrics and error metrics. Furthermore, our recommender system is not designed to predict ratings but to provide a ranked list of items. As a consequence only accuracy metrics are suitable to evaluate our approach. We therefore decided to use the precision and the recall scores, as the most popular accuracy metrics in the field of recommender systems. The test set contains, for each user and each run, a set of items that the user has actually rated and that are not linked to that user in the training set. The recall score ($recall@n$) is the average (on all users) of the proportion (in percentages) of items from the test set that appear among the *top-n* of the ranked list from the training set, for some given n. This measure should be as

[2] `http://grouplens.org/datasets/movielens/`



(a) FilmTrust dataset

(b) Ciao dataset

(c) MovieLens dataset

(d) Netflix dataset

Fig. 1: The influence of dimensionality reduction on recall and precision scores

high as possible for good performance. A recall score of 100 percent indicates that the method always positions the items in the test set among the *top-n* of the ranked list. The precision score (*precision@n*) is the average (on all users) of the proportion (in percentages) of items in the *top-n* that appear in the test set.

3.2 Impact of Dimensionality Reduction

As described in Section 2, the optimization process can be sped up by limiting the number of decision variables. However, dimensionality reduction also decreases potential useful information. In the present section, we aim at estimating a good balance between the number of decision variables and the quality of recommendations. We present at Fig. 1 the evolution of recall@20 and precision@20 for the FilmTrust dataset (Fig. 1a), the Ciao dataset (Fig. 1b), the MovieLens dataset (Fig. 1c), and the Netflix dataset (Fig. 1d), according to the number of components. The user-based formulation and the item-based

one are distinguished, as well as the two dimensionality reduction techniques applied in this work: Principal Component Analysis and Nonnegative Matrix Factorization.

For the Ciao, the MovieLens, and the Netflix datasets, Fig. 1 shows one general trend. The accuracy (recall and precision scores) increases when the number of principal components increases. However, this gain is limited after a number of components approximately equal to 20. This behavior is very interesting since it allows to greatly reduce the number of decision variables. To illustrate the impact of reduction, let us remind that in the user-based form of the problem, the number of decision variables for the MovieLens dataset is initially equal to $10,681$ (the number of movies) while in the item-based form, it is equal to $71,567$ (the number of users). These same numbers can be reduced to about 20 while providing interesting results in regard to recall@20 and precision@20.

By comparing the user-based and the item-based approaches, we observe that the former generally provides lower recall and precision scores than the latter. Even if this observation is made for the four datasets, we really think that it is dataset-dependent and does not imply that the user-based formulation provides systematically worse results on other datasets. To support that, one can see for the Ciao dataset that the evolution of the recall@20 and the precision@20 is similar for the item-based formulation of the recommender system and the user-based formulation (when PCA is used as a dimensionality reduction technique). An explanation can also be found in (Ning et al, 2015) where the authors consider five criteria when dealing with the implementation of a user-based and an item-based recommender system. For the accuracy criteria, they conclude that when the number of users is much greater than the number of items, item-based methods produce more accurate recommendations. When the system has less users than items, user-based methods should be considered.

Another comparison can be made between the dimensionality reduction techniques. Except for the FilmTrust dataset, recall and precision scores are always better with PCA rather than NMF. As a consequence, we can not conclude that imposing nonnegative values in the reduction process improves the accuracy compared to PCA.

3.3 Additional Constraints

The recall and precision scores presented at Fig. 1 only rely on the optimization of the objectives described by Equations (2) and (3). However, remember that one advantage of our approach is its simplicity to add more constraints in the objective function depending on the context. The only condition is that these additional constraints need to be written as linear least-squares. For instance, we can adapt the objective function by taking into account the information of users (or items) similar to the user (or the item) concerned by the optimization.

Table 1: Comparison between formulations of our recommender system with neighbors and without neighbors

FilmTrust dataset								
Neighbors	User-based				Item-based			
	Recall@20		Precision@20		Recall@20		Precision@20	
	PCA	NMF	PCA	NMF	PCA	NMF	PCA	NMF
Without	66.88	81.31	11.49	13.01	81.86	81.67	13.09	13.06
With	**70.55**	**81.33**	**11.88**	**13.02**	**81.96**	**81.72**	**13.13**	**13.10**
Ciao dataset								
Neighbors	User-based				Item-based			
	Recall@20		Precision@20		Recall@20		Precision@20	
	PCA	NMF	PCA	NMF	PCA	NMF	PCA	NMF
Without	**14.82**	12.63	**1.80**	1.56	**14.85**	13.70	**1.80**	1.67
With	13.15	**14.80**	1.70	**1.80**	14.49	**14.08**	1.78	**1.72**
MovieLens dataset								
Neighbors	User-based				Item-based			
	Recall@20		Precision@20		Recall@20		Precision@20	
	PCA	NMF	PCA	NMF	PCA	NMF	PCA	NMF
Without	26.15	23.22	12.67	11.74	31.19	29.45	**14.16**	13.30
With	**31.77**	**25.41**	**15.44**	**12.11**	**31.25**	**29.55**	14.13	**13.39**
Netflix dataset								
Neighbors	User-based				Item-based			
	Recall@20		Precision@20		Recall@20		Precision@20	
	PCA	NMF	PCA	NMF	PCA	NMF	PCA	NMF
Without	13.07	11.52	10.99	10.52	20.85	**18.66**	14.12	12.59
With	**13.57**	**12.77**	**11.72**	**11.15**	**20.90**	18.04	**14.35**	**12.94**

This is the first case studied in the present section. The second case concerns the popularity of items.

Remember that the previous section shows that the number of decision variables can be reduced to around 20, while maintaining comparable accuracy scores (beyond 20 components, the recall@20 is not greatly increased). Therefore, to conduct the next experiments (i.e., for the two cases described in this section), we fixed the number of components for the two dimensionality reduction techniques. The choice was made in order to obtain good results in terms of recall and precision but also to be able to obtain these results as fast as possible: the number of components was fixed to 20 for the FilmTrust dataset, to 30 for the Ciao dataset, to 28 for the MovieLens dataset, and to 30 for the Netflix dataset.

3.3.1 Case 1: the Neighbors

Equation (14) shows how to influence the prediction with neighbors. Here, we aim at evaluating the impact, on accuracy, of including neighbors in our recommender system. For that matter, neighbors were selected by using the cosine similarity measure (which is known to provide good results). For the four datasets and each tested method, we have evaluated the number of neighbors providing the best recall@20. The following numbers of neighbors were

considered: 1, 3, 5, 10, 40, 70, 100, 500, 1000. In Table 1, we present recall@20 and precision@20 obtained for different formulations of our recommender system: user-based formulation with PCA, user-based formulation with NMF, item-based formulation with PCA, and item-based formulation with NMF.

It is clearly impossible to define an optimal number of neighbors, which is highly dependent on the method and the dataset. Nevertheless, the number of neighbors has not a strong influence on the computing time since the neighbors are only used to compute a mean in an adapted version of the Mahalanobis distance.

An interesting advantage is deduced from the analysis of Table 1. In Section 3.2, it seemed that the best formulation of our approach was the item-based one. In Table 1, the best recall@20 and precision@20 for the MovieLens dataset are obtained by the user-based formulation. Another trend of Section 3.2 was that using NMF as a dimensionality reduction technique provided lower recall and precision scores than using PCA. Table 1 illustrates that this conclusion has to be mitigated. For the Ciao dataset, the user-based formulation with NMF and neighbors provides similar results than the item-based formulation with PCA and no neighbors. As a consequence, it allows to choose the formulation with fewer decision variables: if the number of users is greater than the number of items in a dataset, the user-based approach will be preferred for this dataset.

Finally, the general trend of Table 1 is that including neighbors often improves the recall and precision scores for the four datasets.

3.3.2 Case 2: The Popularity

The general optimization model defined at Equation (7) seeks to maximize the preference of users for items. However, this strategy could not be totally appropriate since for some users, diversity is an important parameter of satisfaction. Additional criteria can be added in the optimization model to help preserving some diversity in the recommendations.

One way to achieve this is to penalize the popularity of items in the recommender system, as shown in this section for the user-based formulation of our system. We first evaluate the popularity of each item, as the mean of the ratings given by all the users for this item. Let us note μ_i the popularity of the item i. Then, for each user, we compute his trend to like popular items. For this purpose, we extract the popularity μ_i of each item rated by a user j and we compute the mean ν_j of these popularity scores. As a consequence, the additional constraint can only concern a proportion of users: e.g., the 10% of users with the lowest popularity score ν_j. The proportion of users concerned by this additional constraint is noted n_u. Let us remind that the additional constraint needs to be written with a norm. The adapted optimization model (without applying dimensionality reduction technique) for the user-based case is:

$$\min_{\mathbf{x}_j} \frac{1}{2} \left\| \mathbf{L}^{\mathbf{T}} \mathbf{x}_j - \mathbf{L}^{\mathbf{T}} \bar{\mathbf{x}} \right\|_2^2 + \frac{1}{2} \sum_{i \in \kappa_j} (x_{ij} - m_{ij})^2 + \frac{1}{2} \left\| \boldsymbol{\mu} \mathbf{x}_j \right\|_2^2 \quad , \tag{18}$$

where $\boldsymbol{\mu}$ is a row vector characterizing the popularity μ_i of each item.

To evaluate the impact of the constraint penalizing popular items, we compute the popularity score μ_i of each item i and we retain the *top*-100. Then, the idea is to observe if these popular items are often recommended to the users. To do so, we use the user-based formulation of our recommender system and apply PCA as dimensionality reduction technique. For each ranking position (from rank 1 to rank 20 where rank 1 is related to the first recommended item and rank 20 is related to the twentieth recommended item), we compute the number of users for which a popular item in the *top*-100 has been recommended. As we explained, the constraint defined at Equation (18) can only concern a proportion of users n_u. Therefore, comparisons are made at Fig. 2 for values of n_u varying from 0 to 1. Since our intention is not to improve the accuracy of the method but to show how to add constraints easily into the problem, experiments were made only on the FilmTrust, Ciao, and MovieLens datasets.

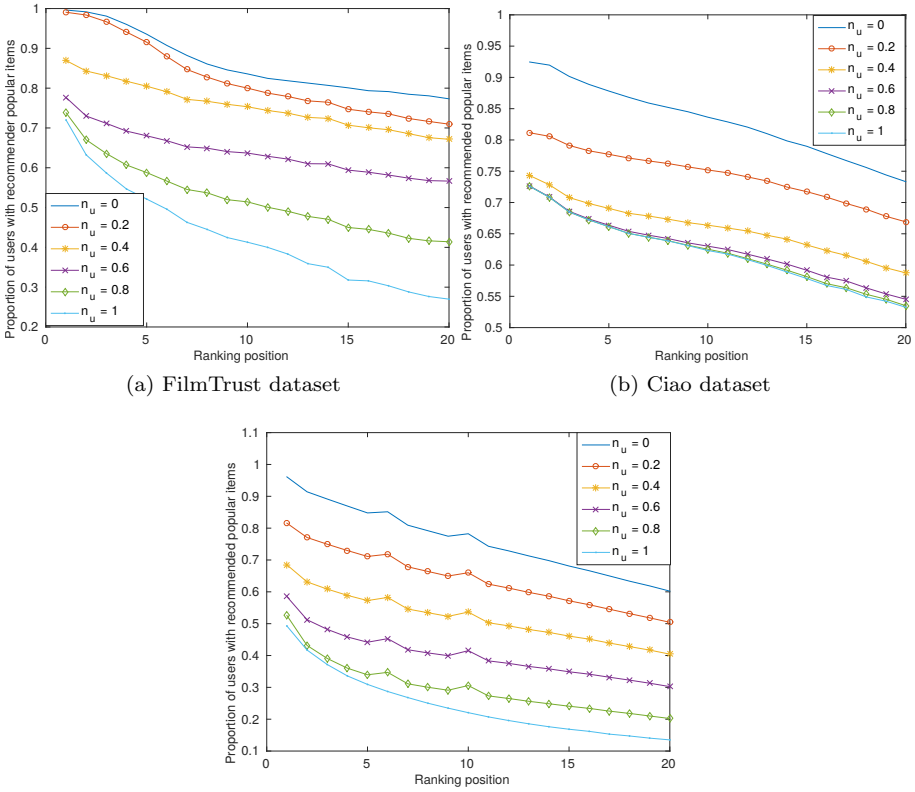
Fig. 2 represents, for each ranking position, the proportion of users for which a popular item in the *top*-100 is recommended. When n_u is equal to 0, it means that no user is concerned by the constraint penalizing popular items. If we observe the curve corresponding to $n_u = 0$ in Fig. 2a, Fig. 2b and Fig. 2c, we notice that the algorithm tends to promote popular items since at rank 1 of recommendations, a popular item is recommended to more than 90% of the users. Increasing the value of n_u has the same influence among the three datasets but not with the same strength. When 20% of the users are concerned by the constraint penalizing popular items, a popular item is recommended at rank 1 to still more than 90% of the users for the FilmTrust dataset, to more than 80% of the users for the Ciao and the MovieLens datasets. A similar trend is observed for the other ranking positions.

Fig. 2 shows that the constraint defined at Equation (18) has a strong influence on the popularity of recommended items. However, penalizing the popularity has an influence on the recall and precision scores. If the main purpose of a recommender system is to propose diversity in the recommendations, the challenge is to find a satisfying trade-off between accuracy (good recall and precision) and utility (limiting the recommendation of popular items).

3.4 Discussion

In previous sections, the accuracy of the proposed algorithms and the influence of additional constraints are analyzed. In the present section, we discuss the special features of the proposed algorithms and make comparisons with some methods of the literature.

For the user-based formulation of our model, recommendations are obtained by solving a convex model for which the number of decision variables



(a) FilmTrust dataset

(b) Ciao dataset

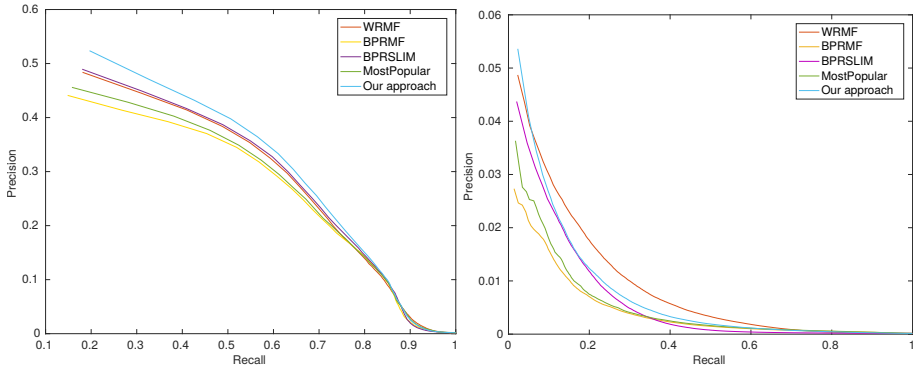
(c) MovieLens dataset

Fig. 2: Proportion of users with recommended popular items at each ranking position

depends on the number of items. For the item-based formulation, the number of decision variables depends on the number of users. Therefore, the choice between the user-based and the item-based formulations can be made depending on these parameters. The item-based formulation will be preferred if the dataset has very few users compared to items, while the user-based formulation will be preferred if the dataset has very few items compared to users.

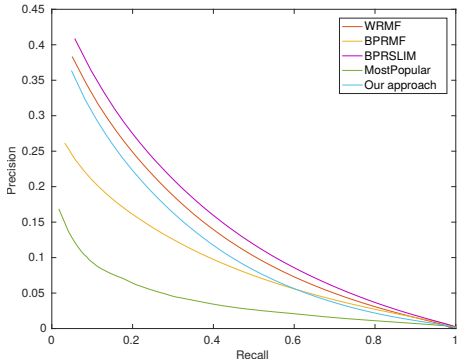
In order to make comparisons of the accuracy results with some state-of-the-art methods based on matrix factorization, we have extracted the best results obtained in the previous sections for the user-based and the item-based formulations of our recommender system. Before presenting results, we briefly describe the methods used for comparison.

– A simple scoring algorithm is the maximum-frequency algorithm (MaxF or MostPopular). It simply ranks the items by the number of users who rated them. In other words, items are suggested to each user in order of decreasing popularity. The ranking is thus the same for all the users.

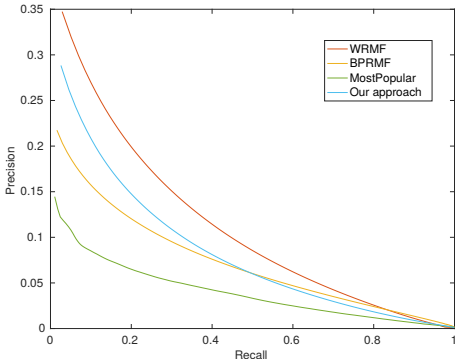


(a) FilmTrust dataset

(b) Ciao dataset



(c) MovieLens dataset



(d) Netflix dataset

Fig. 3: Precision-recall curves for different approaches

The maximum-frequency algorithm serves as a reference to appreciate the quality of the methods.

– A method based on weighted matrix factorization is proposed in (Hu et al, 2008). The authors did not try to predict a rating but they worked with implicit feedback. They treated the data as indication of positive preference, just as we do in the present work. In the following, this method is referred as WRMF (Weighted Regularized Matrix Factorization) method.
– Still in the context of item recommendation based on implicit feedback, authors in (Rendle et al, 2009) presented a generic optimization criterion for personalized ranking which is derived from a Bayesian analysis of the problem. In the following, this method is referred as BPRMF.
– In the recommender system library MyMediaLite (Gantner et al, 2011), developers extended the BPRMF method using the optimization method described in (Ning and Karypis, 2011). In the following, this method is referred as BPRSLIM.

Table 2: Values of the parameters used for WRMF, BPRMF, and BPRSLIM

Method	Dataset	Parameters			
		regularization		alpha	
WRMF	FilmTrust	10		10	
	Ciao	10		10	
	MovieLens	0.001		1	
	Netflix	0.1		1	
		reg_u	reg_i	reg_j	learn_rate
BPRMF	FilmTrust	10	0.0001	0.1	0.01
	Ciao	0.01	10	0.0001	0.01
	MovieLens	0.0001	0.0001	0.001	0.1
	Netflix	0.001	0.0001	0.001	0.1
		reg_i	reg_j	learn_rate	
BPRSLIM	FilmTrust	0.001	1	0.01	
	Ciao	0.0001	0.0001	0.1	
	MovieLens	0.0001	1	0.01	

Let us note that the aforementioned methods were implemented using the recommender system library MyMediaLite (Gantner et al, 2011). For methods based on matrix factorization (WRMF and BPRMF), we chose the same number of components than for our approach, i.e., 20 components for the FilmTrust dataset, 30 components for the Ciao dataset, 28 components for the MovieLens dataset, and 30 components for the Netflix dataset. The other parameters (regularization parameters and learning rates) of each approach were fixed by choosing the value providing the best NDCG for the first test set among these values: 0.0001, 0.001, 0.01, 0.1, 1, 10, 100. These chosen values are presented in Table 2.

In order to compare those methods with our approach, we plotted on Fig. 3 precision-recall curves for all the methods and all the datasets. Concerning the Netflix dataset, BPRSLIM did not provide a solution after more than 24 hours. Given the number of parameters and the fact that the experiments are performed on 10 test sets, we decided not to continue experiments for this method on this dataset. We refer the reader to (Shani and Gunawardana, 2011) for more information on precision-recall curves. What is important to keep in mind is that the best method have the uppermost precision-recall curve on the graph.

Logically, the method based on MaxF (MostPopular) obtains among the worst results. On the FilmTrust dataset, we observe that method based on MaxF (MostPopular) is not that bad. It can be explained by the fact that for this dataset, recommending very popular items is efficient, as suggested by Fig. 2a.

If we look at the two best methods for each dataset, we have:

- FilmTrust dataset: Our approach and BPRSLIM.
- Ciao dataset: WRMF and our approach.
- MovieLens dataset: BPRSLIM and WRMF.
- Netflix dataset: WRMF and our approach.

These observations are also made when evaluating the Normalized Discounted Cumulative Gain (NDCG) for the different datasets. NDCG is a popular metric for evaluating ranked results (Järvelin and Kekäläinen, 2002). We computed NDCG in the same way as in the MyMediaLite library. Table 3 provides values of NDCG obtained for the different methods on the four datasets.

Table 3: Values of NDCG (in %) obtained for the different datasets. Values for the two best methods are in bold.

	MostPopular	BPRSLIM	BPRMF	WRMF	Our approach
FilmTrust	63.25	**65.15**	62.31	65.14	**66.73**
Ciao	18.86	19.23	18.34	**21.38**	**20.37**
MovieLens	35.88	**53.00**	45.44	**50.87**	49.05
Netflix	36.56	/	42.44	**48.03**	**45.16**

Clearly, Fig. 3 and Table 3 state that our approach is very competitive with other methods of the literature based on matrix factorization in the context of item recommendation with implicit feedback. It is also interesting to evaluate the complexity of our solution. In Table 4, computational and space complexities are provided for the 4 main steps of the proposed approach. This analysis is made for the user-based formulation and considering NMF as a reduction technique. With another dimensionality reduction technique, only the computational complexity of the first step needs to be adapted (but clearly, PCA computed with SVD is not a good candidate for very large datasets). The following parameters are used in Table 4: K is the number of known elements in matrix $\mathbf{M}$, r is the number of components provided by matrix factorization, m is the number of rows of matrix $\mathbf{M}$, and n is the number of columns of matrix $\mathbf{M}$. In order to illustrate the impact of the computational complexity, the last column of Table 4 provides the execution times for each of the steps when performing our approach on the largest dataset, the Netflix one.

Obviously, matrix factorization is the most time-consuming part of the approach. HALS algorithm used for factorizing matrix $\mathbf{M}$ has a o(Kr) computational complexity (Gillis and Glineur, 2012). Other steps of the algorithm are performed on small matrices (typically $(r \times r)$) such as $\mathbf{\Sigma^{-1}}$ and $\mathbf{L_{\Phi}}$. Finally, let us note that the last step of the algorithm needs to be performed for each user. The other steps are performed only once for all the users.

Concerning the execution times of Table 4, tests were performed on an Intel Xeon E5-2630 v3 (2.40GHz). In real applications involving a recommender system, the recommendations are provided to one user at a time. For the user-based formulation of our approach, the recommendations for one user can be computed online (given the results presented at Table 4). For the item-based formulation, the recommendations can be computed offline and then provided online in real-time to a user. As a consequence, our approach is transposable to a real practice.

Table 4: Computational and space complexities of the proposed approach

Step	Comput. Complexity	Space Complexity	Exec. Time (s)
Factorizing $\mathbf{M}$ with HALS algorithm	o(Kr)	$\mathbf{M}$ is a sparse matrix with K elements $\mathbf{\Phi}$ is a $(m \times r)$ matrix $\mathbf{B}$ is a $(r \times n)$ matrix	97.4
Computing the inverse covariance matrix $\mathbf{\Sigma^{-1}}$ on $\mathbf{B}$	o(r^3)	$\mathbf{\Sigma^{-1}}$ is a $(r \times r)$ matrix	9.4×10^{-3}
Decomposing $\mathbf{\Sigma^{-1}}$ with Cholesky method to obtain $\mathbf{L_\Phi}$	o(r^3)	$\mathbf{\Sigma^{-1}}$ and $\mathbf{L_\Phi}$ are $(r \times r)$ matrices	5.2×10^{-5}
Using operator \ to solve Equation (13)	o(r^3)	Equation (13) involves $\mathbf{L_\Phi}$ and the rows of $\mathbf{\Phi}$ related to items already considered by the user	8.1×10^{-4}

4 Conclusion

This paper suggests a convex optimization model to produce recommendations, which is adaptable, fast, and scalable - while remaining very competitive to state-of-the-art methods in terms of accuracy.

The **adaptability** of the model refers to two different dimensions. Firstly, the model can be viewed from a user-based perspective or from an item-based perspective, allowing to choose, depending on the context, the formulation with fewer decision variables. Secondly, additional constraints, depending on the context, can easily be added to the optimization model. In this paper, two cases are presented. In the first one, we added a constraint so that a group of very close neighbors (of a user or an item) has a stronger influence on the recommendations. We showed through experiments performed on four datasets of different sizes (i.e., FilmTrust, Ciao, MovieLens, and Netflix) that this constraint can improve the accuracy of the recommendations. The second additional constraint shown in this paper tries to penalize popular items and thus helps preserving diversity in the recommendations. We showed that the proportion of users with recommended popular items at each ranking position can be greatly reduced.

Fast and **scalable** as the number of decision variables can be drastically reduced by coupling our optimization model with matrix factorization techniques, such as the two ones considered in this paper: Principal Component Analysis and Nonnegative Matrix Factorization. Experiments have highlighted that only about 20 to 30 components computed by matrix factorization are useful in the datasets to get accurate results. This trend is valid both for the user-based and the item-based formulations of our approach, regardless the number of constraints. As a consequence, average computing time to solve the optimization problem is less than $1ms$ in general, for 1 user/item.

Accurate as we have compared our approach to state-of-the-art methods based on matrix factorization, showing that our approach is very competitive regarding the precision-recall curve and NDCG metric.

In the future, it could be interesting to link additional constraints based on rich objects such as multimedia objects. Our system could be thereby designed for social media-driven personalization. Therefore, the most challenging task will be to write these constraints with a convex form.

Acknowledgements The authors would like to thank Marco Saerens, Nicolas Gillis, and Arnaud Vandaele for insightful comments on this work.

References

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on 17(6):734–749

Agarwal D, Chen BC, Elango P, Wang X (2011) Click shaping to optimize multiple objectives. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11), pp 132–140

Armstrong J (2001) Principles of Forecasting, A Handbook for Researchers and Practitioners. Kluwer Academic

Bennett J, Lanning S (2007) The Netflix prize. In: KDD Cup and Workshop in conjunction with KDD

Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge University Press

Cremonesi P, Koren Y, Turrin R (2010) Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10), pp 39–46

Devooght R, Kourtellis N, Mantrach A (2015) Dynamic matrix factorization with priors on unknown values. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 189–198

Deza M, Deza E (2014) Encyclopedia of distances, 3rd edn. Springer

Feldman S (2012) Multi-Task Averaging: Theory and Practice. PhD thesis, University of Washington

Fouss F, Francoisse K, Yen L, Pirotte A, Saerens M (2012) An experimental investigation of kernels on graphs for collaborative recommendation and semi-supervised classification. Neural Networks 31:53–72

Fouss F, Saerens M, Shimbo M (2016) Algorithms and Models for Network Data and Link Analysis. Cambridge University Press

Gantner Z, Rendle S, Freudenthaler C, Schmidt-Thieme L (2011) MyMediaLite: A free recommender system library. In: Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011)

Gill PE, Murray W, Wright MH (1981) Practical optimization. Academic Press

Gillis N (2011) Nonnegative Matrix Factorization: Complexity, Algorithms and Applications. PhD thesis, Université catholique de Louvain

Gillis N (2014) The Why and How of Nonnegative Matrix Factorization. ArXiv e-prints URL http://adsabs.harvard.edu/abs/2014arXiv1401.5226G, 1401.5226

Gillis N, Glineur F (2012) Accelerated multiplicative updates and hierarchical Als algorithms for nonnegative matrix factorization. Neural Computation 24(4):1085–1105

Gu Q, Zhou J, Ding C (2010) Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In: Proceedings of the 2010 SIAM International Conference on Data Mining, pp 199–210

Guo G, Zhang J, Yorke-Smith N (2013) A novel bayesian similarity measure for recommender systems. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI), pp 2619–2625

Herlocker J, Konstan J, Riedl J (2002) An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Information Retrieval 5:287–310

Hill W, Stead L, Rosenstein M, Furnas G (1995) Recommending and evaluating choices in a virtual community of use. Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems pp 194–201

Hsieh CJ, Olsen P (2014) Nuclear norm minimization via active subspace selection. In: Proceedings of the 31st International Conference on Machine Learning (ICML-14), pp 575–583

Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM), pp 263–272

Jambor T, Wang J (2010) Optimizing multiple objectives in collaborative filtering. In: Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10), pp 55–62

Järvelin K, Kekäläinen J (2002) Cumulated gain-based evaluation of ir techniques. ACM Trans Inf Syst 20(4):422–446

Johnson R, Wichern D (2002) Applied multivariate statistical analysis, 5th edn. Prentice Hall

Kannan R, Ishteva M, Park H (2012) Bounded matrix low rank approximation. In: Proceedings of the 12th IEEE International Conference on Data Mining (ICDM), pp 319–328

Kim D, Yum BJ (2005) Collaborative filtering based on iterative principal component analysis. Expert Systems with Applications 28(4):823 – 830

Koren Y (2008) Factorization meets the neighborhood: A multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08), pp 426–434

Koren Y (2009) Collaborative filtering with temporal dynamics. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09), pp 447–456

Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37

Lilien G, Smith B, Moorthy K (1992) Marketing Models. Prentice Hall

Lü L, Medo M, Yeung CH, Zhang YC, Zhang ZK, Zhou T (2012) Recommender systems. Physics Reports 519:1–49

Luo X, Zhou M, Xia Y, Zhu Q (2014) An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. Industrial Informatics, IEEE Transactions on 10(2):1273–1284

Murthi B, Sarkar S (2003) The role of the management sciences in research on personalization. Management Science 49(10):1344–1362

Ning X, Karypis G (2011) SLIM: Sparse linear methods for top-n recommender systems. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp 497–506

Ning X, Desrosiers C, Karypis G (2015) A Comprehensive Survey of Neighborhood-Based Recommendation Methods, Springer US, pp 37–76

Paterek A (2007) Improving regularized singular value decomposition for collaborative filtering. Proceedings of KDD Cup and Workshop pp 39–42

Rao N, Yu HF, Ravikumar PK, Dhillon IS (2015) Collaborative filtering with graph information: Consistency and scalable methods. In: Advances in Neural Information Processing Systems 28, Curran Associates, Inc., pp 2107–2115

Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (AUAI), pp 452–461

Resnick P, Neophytos I, Mitesh S, Bergstrom P, Riedl J (1994) GroupLens: An open architecture for collaborative filtering of netnews. Proceedings of the Conference on Computer Supported Cooperative Work pp 175–186

Rich E (1979) User modeling via stereotypes. Cognitive Science 3(4):329–354

Rodriguez M, Posse C, Zhang E (2012) Multiple objective optimization in recommender systems. In: Proceedings of the Sixth ACM Conference on Recommender Systems (RecSys '12), pp 11–18

Salakhutdinov R, Mnih A (2008) Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In: Proceedings of the 25th International Conference on Machine Learning (ICML '08), pp 880–887

Salton G (1989) Automatic Text Processing. Addison-Wesley

Shan H, Banerjee A (2010) Generalized probabilistic matrix factorizations for collaborative filtering. In: Proceedings of the 10th IEEE International Conference on Data Mining (ICDM), pp 1025–1030

Shani G, Gunawardana A (2011) Evaluating recommendation systems. Recommender Systems Handbook pp 257–297

Shardanand U, Maes P (1995) Social information filtering: Algorithms for automating 'word of mouth'. Proceedings of the Conference on Human Factors in Computing Systems pp 210–217

Smola AJ, Kondor R (2003) Kernels and regularization on graphs. In: Learning Theory and Kernel Machines, Lecture Notes in Computer Science, vol 2777,

Springer Berlin Heidelberg, pp 144–158
Tang J, Gao H, Liu H (2012) mTrust: Discerning multi-faceted trust in a connected world. In: Proceedings of the fifth ACM international conference on Web search and data mining, pp 93–102
Yu K, Zhu S, Lafferty J, Gong Y (2009) Fast nonparametric matrix factorization for large-scale collaborative filtering. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09), pp 211–218
Zhang S, Wang W, Ford J, Makedon F (2006) Learning from incomplete ratings using non-negative matrix factorization. In: Proceedings of the 2006 SIAM International Conference on Data Mining, pp 549–553
Zhang Y, Chen W, Yin Z (2013) Collaborative filtering with social regularization for TV program recommendation. Knowledge-Based Systems 54:310–317