Expert Systems with Applications 42 (2015) 4733–4742

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Transforming collaborative filtering into supervised learning

Filipe Braida [a,b,*,1], Carlos E. Mello [b,1], Marden B. Pasinato [a,2], Geraldo Zimbrão [a,2]

[a] PESC/COPPE, Universidade Federal do Rio de Janeiro, Cidade Universitária, Rio de Janeiro, P.O. Box: 68511, Brazil
[b] DCC/IM, Universidade Federal Rural do Rio de Janeiro, Nova Iguaçu, Rio de Janeiro 26020-740, Brazil

ARTICLE INFO

Article history:
Available online 17 January 2015

Keywords:
Recommender system
Collaborative filtering
Dimensionality reduction
Supervised learning

ABSTRACT

Collaborative Filtering (CF) is a well-known approach for Recommender Systems (RS). This approach extrapolates rating predictions from ratings given by user on items, which are represented by a user-item matrix filled with a rating $r_{i,j}$ given by an user i on an item j. Therefore, CF has been confined to this data structure relying mostly on adaptations of supervised learning methods to deal with rating predictions and matrix decomposition schemes to complete unfilled positions of the rating matrix. Although there have been proposals to apply Machine Learning (ML) to RS, these works had to transform the rating matrix into the typical Supervised Learning (SL) data set, *i.e.*, a set of pairwise tuples (x, y), where y is the correspondent class (the rating) of the instance $x \in \mathbb{R}^k$. So far, the proposed transformations were thoroughly crafted using the domain information. However, in many applications this kind of information can be incomplete, uncertain or stated in ways that are not machine-readable. Even when it is available, its usage can be very complex requiring specialists to craft the transformation. In this context, this work proposes a domain-independent transformation from the rating matrix representation to a supervised learning dataset that enables SL methods to be fully explored in RS. In addition, our transformation is said to be straightforward, in the sense that, it is an automatic process that any lay person can perform requiring no domain specialist. Our experiments have proven that our transformation, combined with SL methods, have greatly outperformed classical CF methods.

1. Introduction

The growth of Internet has brought a myriad of new business and applications by reshaping many human activities. For instance, there has been a significant shift in trading towards more flexibility, availability, and mobility. A million of e-commerce systems around the globe have been deployed providing all these features and offering their users an unlimited catalog of goods and services. Due this overabundance of possibilities, users may become anxious and abandon the purchase process, once making choices with too many options may be bothersome and difficult (Schwartz, 2005).

In this scenario, Recommender Systems (RS) have emerged playing the important role of providing users with recommendations. These systems have been widely adopted by big players of the Web such as Amazon[3], NetFlix[4], YouTube[5] and many others. Besides their obvious virtues of leveraging sales, RS also improve customer loyalty and increase cross-sales. In fact, these systems try to efficiently meet needs and interests of users by avoiding the burden of finding a needle in a haystack (Schafer, Konstan, & Riedi, 1999).

The most prominent and successful RS approach is Collaborative Filtering (CF). It aims to recommend an item to a user based only on item evaluations (often represented by a numerical rating) provided by other users in the system. This scheme may be modeled through a $n_{users} \times m_{items}$ matrix R, where each position $r[i,j]$ of R is filled with the corresponding rating value $r_{i,j}$ given by an user i to an item j. Often, the rating matrix is sparse, in the sense that, very few ratings are known. The positions with unknown ratings are conventionally filled with a nonexistent rating symbol $\varnothing$.

Accordingly, CF algorithms predict ratings for the unfilled positions (unknown ratings) by extrapolating from the filled ones in the rating matrix. Thus, each unfilled matrix position can be viewed as an instance that needs to be classified into one of the

* Corresponding author at: PESC/COPPE, Universidade Federal do Rio de Janeiro, Cidade Universitária, Rio de Janeiro, P.O. Box: 68511, Brazil. Tel.: +55 21 2562 8672.
E-mail addresses: filipebraida@ufrj.br (F. Braida), carlos.mello@ufrrj.br (C.E. Mello), marden@ufrj.br (M.B. Pasinato), zimbrao@cos.ufrj.br (G. Zimbrão).
[1] Tel.: +55 21 2669 0105.
[2] Tel.: +55 21 3938 8672.

[3] http://www.amazon.com.
[4] http://www.netflix.com.
[5] http://www.youtube.com.

http://dx.doi.org/10.1016/j.eswa.2015.01.023
0957-4174/

possible ratings. Note that, no other input data than the rating matrix is required in this approach.

The CF data scheme creates a barrier for applying Supervised Learning (SL) as it is not possible to represent ratings as points in a vector space $\mathbb{R}^k$, like in classification and regression tasks. Indeed, there are no straightforward features to build an input space from which SL methods could learn rating prediction models.

There have been proposals that attempt to transform the rating matrix into a typical SL dataset suitable for ML methods. Nevertheless, the proposed transformations rely on the domain information, which can be hard to extract and even misleading (Cunningham & Smyth, 2010; Hsu, Wen, Lin, Lee, & Lee, 2007; O'Mahony & Smyth, 2009, 2010).

Moreover, a transformation process based on available domain information is still a complex task, once only specialists are able to execute. The transformations proposed so far are thoroughly crafted using the domain information and are also very specific. Thus one may claim that no straightforward domain-independent methodology has been proposed to transform the CF task into a SL task.

In this context, this work proposes a novel straightforward domain-independent methodology to transform the CF scheme into a SL scheme, in the sense that, this is an automatic process which only requires the rating matrix as input. The positions of the rating matrix R are mapped into a k-dimensional vector space corresponding to the k most important latent factors of R. Each point in this feature space is associated with its corresponding rating value forming a typical SL dataset. SL methods are trained with this dataset and applied to predict the users' unknown ratings. Their performance was given according to metrics such as MAE and RMSE, which indicate that this approach has greatly outperformed classical CF techniques.

This paper is organized in 5 sections of which this is the first one. Section 2 presents related work on how to apply supervised learning techniques in recommender systems. In Section 3, the general proposal is described in details and some theoretical insights are provided. Along Section 4, the experimental settings and results are presented. Finally, a discussion about the weakness and strengths of our proposal is carried out in Section 5 as well as some future works.

2. Related work

Collaborative Filtering (CF) approach has been largely used in Recommender Systems (RS). This approach aims to extrapolate rating predictions from a user-item matrix filled with the corresponding ratings given by users to items. The main related issue of this approach relies on the sparsity of this rating matrix. To handle this, many CF algorithms have been proposed based on adaptations of classic classification and regression methods (Ricci, Rokach, Shapira, & Kantor, 2011). For instance, the User-based CF algorithm is a k-Nearest Neighbors classifier with similarity measures between users adapted to consider only the co-rated items in the computation (Adomavicius & Tuzhilin, 2005).

Alternatively, there have been many works that apply dimensionality reduction techniques so as to transform the sparse matrix into a fixed space of features. Sarwar, Karypis, Konstan, and Riedl (2000) proposes the use of the Singular Value Decomposition (SVD) and the recommendations are generated by operations between the resulting matrices.

CF methods based on Matrix Factorization (MF) techniques have received great attention after NetFlix Prize. In this challenge, one of the top accurate methods was based on the factorization of the rating matrix through SVD (Koren, Bell, & Volinsky, 2009; Salakhutdinov & Mnih, 2008b). In addition, this sort of technique has shown to be scalable and accurate even under high sparsity. The idea behind these techniques is to complete the rating matrix through a low-rank approximation (Koren, 2008; Koren et al., 2009; Paterek, 2007). There have been also proposals that exploits, instead of matrix factorization, a probabilistic latent variable framework, wherein hidden random variables constitutes the underlying users' preferences on items (Salakhutdinov & Mnih, 2008a, 2008b).

Another approach to try to deal with the sparsity problem is to treat it as classification/regression problem. An approach of this kind is to derive features from the rating matrix based on the domain knowledge (Cunningham & Smyth, 2010; Hsu et al., 2007; O'Mahony & Smyth, 2009, 2010). For instance, the users' mean of ratings and the standard deviation may constitute features to train classifiers.

In Billsus (1998), authors propose a domain-independent method in order to predict ratings. For each user a model is induced based on a matrix of features derived from the original rating matrix. Although this proposal performs well, it does not work for multiclass problems and is not scalable, since the number of models to learn increases with new users. This method uses SVD to reduce dimensionality and build a new feature space in order to train SL models.

To the best of our knowledge there is no work that proposes a general straightforward domain-independent transformation of the rating matrix into a classic training set for applying SL methods.

3. Proposal

In this section we start discussing the main issue in Collaborative Filtering (CF) that prevents the direct application of Supervised Learning (SL) methods to rating predictions. We follow by describing the proposed methodology of this work.

In CF, the raw dataset consists of ratings $r_{i,j}$ associated with their corresponding user-item pairs, *i.e.* a numerical evaluation $r_{i,j} \in \mathbb{R}$ given by a user i on an item j. Hence, CF only uses information about how users like or dislike certain items. This is often represented by a rating matrix R, where each line i corresponds to a user and each column j corresponds to an item (Adomavicius & Tuzhilin, 2005).

CF aims to learn users' preferences from the matrix R in order to extrapolate rating predictions for items not yet rated by the users, *i.e.* to complete the unfilled positions in R. Usually, the rating matrix is subject to high sparsity, since only few users usually provide ratings by leaving many matrix positions unfilled. This makes the CF task even harder, once the more sparse the rating matrix, the harder the learning and more unfilled positions should be completed with predicted ratings.

The main issue that differentiates the CF problem from the classic SL problem lies in the fact that there is only the rating data from which one may learn a model for the users' preferences. A rating dataset is fundamentally unlike classic SL datasets, where instances are defined in an input feature space and are associated with outputs, in case, ratings. In fact, there is no such well-defined input feature space in a CF scheme so that one may learn a mapping from instances in a domain of features to output values (ratings). Instead, there are user-item pairs that determine the rating values, which are not represented by features.

For applying SL methods to handle CF rating predictions there should be a feature vector associated with each rating in the CF dataset. Each user-item pair in CF data must be transformed into an instance vector of features and labeled with its corresponding rating. In this form, one could deal with this set of labeled instances as a classic supervised training set from which SL methods may be applied.

Although many CF algorithms have been proposed in the literature of recommender systems (Ricci et al., 2011), to the best of our knowledge, there is no clear methodology or method that

aims to transform CF data into a SL training set in order to apply SL methods to predict ratings. The main reason behind this approach is to benefit from a wide range of well-established supervised learning methods and techniques available in the literature of machine learning that are ready to be used.

In this context, we propose a methodology to transform the CF data into a SL dataset by means of building a new input feature space where user-item pairs are represented by vectors of features, labeled with their corresponding ratings. Thus this methodology generates supervised training sets so that SL methods are straightforwardly applicable to enhance rating predictions.

Namely COFILS – *Collaborative Filtering to Supervised Learning*, the proposed methodology exploits the underlying users' preferences through the analysis of latent variables on the rating matrix. The objective is from the extraction of latent variables creating features that define an input space. In this new space one is able to apply classification methods, like Logistic regression, so as to predict ratings for the user-item pairs not yet rated in the original rating matrix representation.

In Fig. 1 the methodology scheme is illustrated. Note that, COFILS is divided into two steps. The first consists in extracting latent variables from the rating matrix. By holding the extracted latent variables, the second step is to build a supervised training set of instances defined by features based on these latent variables and labeled with their corresponding ratings. This SL dataset should allow the induction of classifiers and regressors to perform rating predictions.

The idea behind the use of latent variables is to exploit somehow the underlying users' preferences that explain the outcomes recorded in the rating matrix. There is no clear semantic explanation for latent variables that allows us to interpret users' preferences. Nevertheless, depending on the technique adopted to extract latent variables, one may observe strong interesting hidden relationships that can be taken as features.

In Koren et al. (2009) an analogy with users and movies is described. In this example, a rating matrix representing users' preferences on movies is decomposed into two matrices U (users-by-factors) and V (movies-by-factors). The former is a matrix corresponding to users defined by factors, *i.e.* the extracted latent variables. The latter consists of a matrix describing movies defined by their corresponding extracted factors.

To better understand the role played by the latent variables, one could interpret each factor as an underlying movie category such as comedy, horror, drama, etc. Thus, each user would be defined by latent features (factors) that represent his preferences on each movie category. Analogously, each movie would be assigned to a feature vector that provides how that movie matches each category.

There are many methods for analyzing and extracting latent variable, most of them are based on matrix decomposition/factorization that generates other matrices from which one may benefit for building a new feature space. The majority of the matrix decomposition/factorization methods are from Linear Algebra such as SVD, Tensors, QR, PCA, and others (Koren et al., 2009).

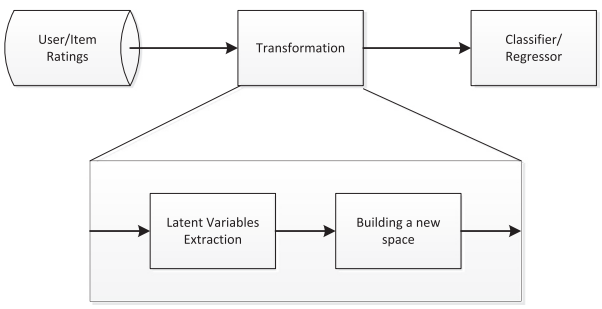

Fig. 1. Procedure to transform collaborative filtering into supervised learning.

After obtaining the latent variables or factors corresponding to each user and item, the methodology builds an input feature space by joining users and items so that a vector of features represents each user-item pair filled in the rating matrix. This last step generates a set of labeled instances, each one representing a user-item pair in a space of features based on latent variables.

In the following subsections, we discuss in details the two step procedure of COFILS. In the next subsection we discuss the extraction of latent variables and how to handle sparsity in the rating matrix. In the posterior subsection we discuss how to properly build the feature space for representing ratings such that SL algorithms can be applied.

3.1. Transformation

3.1.1. Extracting latent variables

In statistics, latent (or hidden) variables are those random variables that are not directly observed. Unlike observable variables, these ones are inferred from mathematical models and observations of other variables. There are many latent variable models that aim to explain observable variables through latent variables.

In CF, latent variables have been used to model the underlying users' preferences from the rating matrix (Koren et al., 2009). To this end, one considers each filled position of the rating matrix as an observation. Thus, users and items are represented by the latent variables that result in the observed ratings.

More specifically, each user i and item j in the rating matrix R are represented by a vector of latent variables (or factors), u_i and v_j, respectively. To obtain back the corresponding rating $r_{i,j}$, one applies u_i and v_j to the latent variable model. Note that, users are in a vector space U such that $u_i \in U$, and items v_j are in a space V such that $v_j \in V$.

Although there are many procedures to extract latent variables, methods based on matrix factorization/decomposition are well-suited for CF, since ratings are originally in that form. However, the main difficulty to apply these methods is dealing with the high sparsity of the rating matrix, *i.e.* a large number of unknown ratings.

A naïve approach is to consider a value out of the range of ratings for the unfilled positions of the rating matrix. For instance, to assign zero value to all unfilled positions, when ratings are from 1 up to 5. The main drawback of this approach is the introduction of bias in the latent variables, once the factorization method considers the zero values to extract the latent variables. Therefore, this approach adds a large bias into the extraction of latent variables, specially in very sparse matrices.

Alternatively, one may consider to fill the unrated positions with the global average of ratings. The main idea behind this approach is to assume that unknown values tend to the rating average. A variation of this approach is to use the average of users' or items' ratings to fill the unknown values. In this case, the drawback lies in dealing with users and items with none or very few ratings.

In Candès and Recht (2009), a very interesting and effective approach is presented to handle the unfilled positions of the rating matrix. To this end, the authors propose to approximate the rating matrix by a low rank matrix in order to apply matrix completion techniques to fill the unknown ratings. This is made by using Single Value Decomposition (SVD), a Linear Algebra technique for matrix factorization based on eigenvectors.

Fig. 2 illustrates the SVD factorization on a rating matrix R. Note that, only two factors (latent variables) are considered, thus the

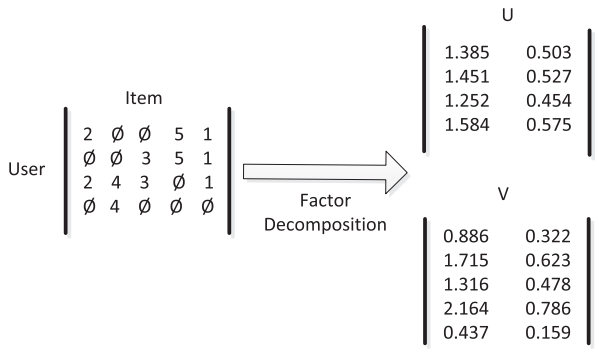

Fig. 2. SVD factorization with 2 factors.

two matrices U and V are obtained with 2 columns each, representing the two factors. In both matrices U and V each row represents an user and an item, respectively. Thus, each user's feature vector u_i is given by the ith row of U and each item's feature vector v_j is given by the jth row of V.

In Fig. 3, the corresponding users (on the left) and items (on the right) are depicted in the latent feature space obtained in the example of Fig. 2. The distribution of users and items in the factor space is given according to their associated ratings in the original matrix. Note that, the closer the users/items are in the latent feature space, the more similar their ratings.

3.1.2. Building a new space

After extracting the latent variables from the rating matrix, we proceed with the second step of the proposed methodology, which consists in building an input feature space where ratings are represented by instances.

To this end, each row of U and V, corresponding to a user u_i and an item v_j, respectively, is organized in order to map the filled positions of the matrix to their corresponding ratings. The objective is to create an instance vector for each position $r[i,j]$ of the matrix R with an associated rating value $r_{i,j}$.

Therefore, each rating of the matrix becomes an instance x defined as

$$x = (u_i, v_j)$$

and labeled with

$$y = r_{i,j}.$$

This mapping results in a set of labeled instances in an input feature space that constitute a supervised training set $D = \{(x,y)_k\}_{k=1}^{n}$, where k corresponds to each position of the original matrix filled with a rating and n is the number of filled positions.

From the supervised training set D in latent feature space, one is able to induce a SL model Θ for predicting ratings through the mapping $\Theta(x) \rightarrow y$. By choosing an appropriate learning model, one may achieve high accuracy in rating predictions. The entire process of COFILS is illustrated in Fig. 4 with a toy example where only one latent variable is used.

In Fig. 5 (left) the toy example is depicted in the input space. Note that, the horizontal axis corresponds to the factor values of users and the vertical axis corresponds to the factor of items. Interestingly, this space is linearly separable, and therefore a simple linear classifier can be used to predict ratings.

The parameters of the proposed methodology are the number of latent variables, the method for extracting latent variables, and the

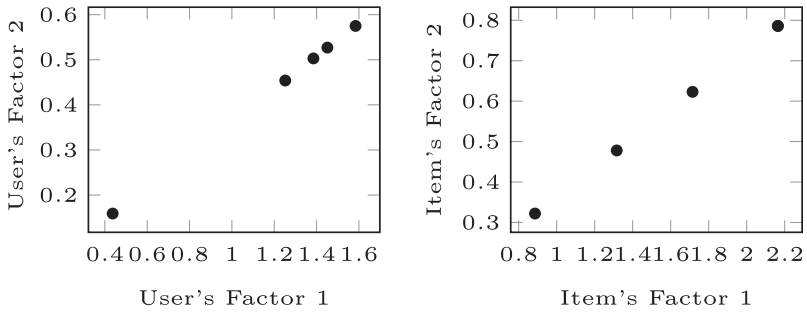

Fig. 3. User (left) and item (right) in factor space.

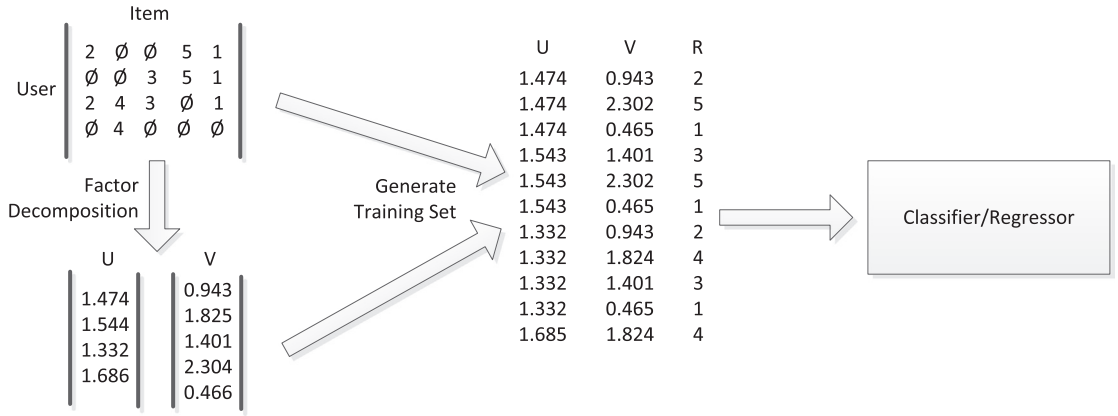

Fig. 4. Procedure for transforming the collaborative filtering problem to supervised learning.

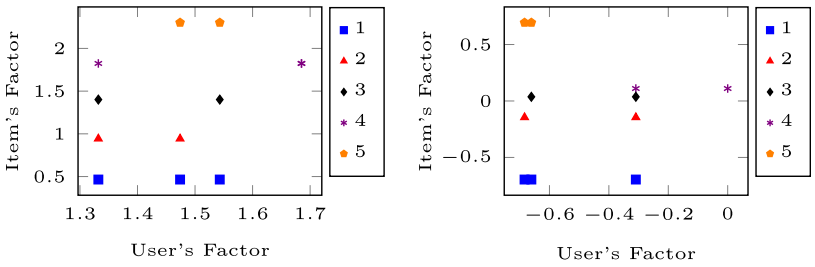

Fig. 5. Training set space generated in the toy example (on the left) and the same example normalized by the average user rating (on the right).

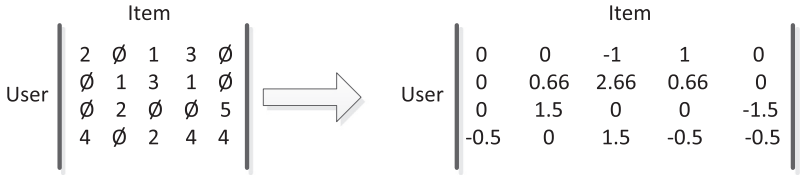

Fig. 6. Preprocessing using the normalization user.

strategy to deal with the unfilled positions of the rating matrix. Each change in the methodology setting may generate different input spaces. Consequently, the performance of the SL model may also change, since the distribution of the instances in the input space changes.

For instance, in Fig. 5, the same toy example is depicted with different strategies to handle the unfilled positions of the rating matrix. On the left hand-side, these positions are filled with the global rating average. On the right hand-side, the unfilled positions are replaced with their corresponding users' average rating, instead of the global average. Note that, despite both pictures comes from the same rating matrix, they lead to different distributions of instances. Therefore, an appropriate choice of the parameters of the methodology may conduct to better SL models for rating prediction.

4. Experiments

In this section we described the experiments conducted in order to evaluate the proposed methodology comparing it with other approaches in the literature. Firstly, we analyze how the number of factors, given by the SVD decomposition, impact our methodology in terms of overall accuracy. Secondly, using the best parameters found in the first experiment, a comparison in terms of overall accuracy is carried out including User-based collaborative filtering, Item-based collaborative filtering, Regularized SVD (Paterek, 2007), Improved Regularized SVD (Paterek, 2007), SVD++ (Koren, 2008; Koren et al., 2009), Bayesian Probabilistic Matrix Factorization (BPMF) (Salakhutdinov & Mnih, 2008a, 2008b) and the classifier/regressor that yielded the best result using our methodology.

4.1. Datasets

Two classic datasets for recommender systems were used in our experiments, both are MovieLens datasets (Adomavicius & Tuzhilin, 2005). The first dataset, MovieLens 100k, comprises 100,000 ratings given by 943 users to 1682 movies. The second dataset, MovieLens 1M, comprises 1,000,209 ratings given by 6040 users to 3952 movies. In both cases, ratings vary from 1 to 5 and demographic information about users and items – such as age, gender, title, release date and others – were discarded.

4.2. Setup

In our experiments, we vary some parameters such as the method used for the latent variable extraction, the number of latent variables and the supervised learning algorithm regarding the overall performance. Among the compared algorithms are k-Nearest Neighbours, Regularized SVD, Improved Regularized SVD, SVD++ and BPMF.

The proposed methodology (COFILS) has three steps: pre-processing, latent variables extraction, and regression/classification. The first step is responsible for representing the data and treating the unfilled positions in the rating matrix. Two approaches were considered: using zero values to fill the matrix, with and without normalization by the average user rating. Those step is shown in Fig. 6.

In the latent variables extraction, a set of three techniques will be used. The first is the Singular Value Decomposition (SVD) using the matrix U as representation of users in latent variables, and the matrix V as representation of item in latent variables. Singular values were discarded (Linden, Smith, & York, 2003). The Improved Regularized SVD technique will also be used as well as the Partial SVD given in LingPipe library.[6]

Three ML methods were chosen to be applied as SL model in the last step of our methodology: Naive Bayes, Artificial Neural Networks (ANNs) and Random Forest. The first was chosen because of its simplicity; the second for its robustness; and the third for its precise accuracy (Caruana & Karampatziakis, 2008). In short, the four step of our methodology can be summarize as follows:

1. Pre-processing: normalization by the average user rating, average item rating or no normalization at all.
2. Latent variable extraction: SVD, Improved Regularized SVD or Partial SVD.
3. Number of latent variables used.
4. SL model: Naive Bayes, ANN or Random Forest.

In addition to these four elements, there are other parameters that need to be set depending on the ML method. For instance, in the case of ANN, we used two hidden layers with the sigmoid

[6] http://alias-i.com/lingpipe/.

function as the activation function. Random Forest has an unique parameter that is the number of decision trees used.

In this work, our goal is to validate our proposal, therefore, we will not carry out an extensive investigation in order to optimally fine-tune these parameters. We are interested only in finding the configuration that allows us to outperform classical CF algorithms.

The first experiment has the purpose of defining the SL model's configuration to be used in the rest of our experiments. In the case of ANN, we will define the number of neurons in the hidden layer. In the Random Forest, we will choose the number of decision trees used. Naive Bayes, on the other hand, requires no parameter to be set.

We decided to use the normalization by the average user rating, in the pre-processing step, and the SVD, in the second step. In order to determine the number of latent variables to build the new feature space, one applies the SVD algorithm on the user-item matrix and analyzes the given singular values on matrix S (see Fig. 7). The idea is to look for small gaps along side the number of latent variables. These gaps suggest relative good cuts on the number of latent variables. Note that in Fig. 7, for small numbers of latent variables one may observe quite large drops, and as one goes further along this axis the curve gets linear. In this way, one suggests a cut on eight latent variables since, from this number of latent variables, the correspondent single values practically cease to decrease. This choice of the number of latent variable is used for both user and item in the first experiment.

The second experiment evaluates the proposed method (COFILS) by varying the four aforementioned parameters. In this way, one should verify how accuracy rate of the SL model varies with the number of latent variables, the method for learning such latent variables, and the pre-processing technique applied to the input matrix. Next, we will show that there is an interesting relationship between these parameters and the performance of the SL model, because the data distribution changes significantly depending on the technique of latent variable extraction and its number.

Once performed the first experiment, a new experiment is conducted so as to compare the proposed method with the main Collaborative Filtering (CF) techniques of the literature. To do so, COFILS parameters are set with the best result parameters obtained in the first experiment, *i.e.*, those parameter values that result in the best performance rates.

Seven CF algorithms: two memory-based and five model-based are taken for comparison. In memory-based CF, variations of the k-Nearest Neighbor algorithm are based on vectors of either users or items, with cosine and Pearson correlation as similarity measures. In model-based CF, Regularized SVD, Improved Regularized SVD, SVD++ and BPMF are considered (Gantner, Rendle, Freudenthaler, & Schmidt-Thieme, 2011).

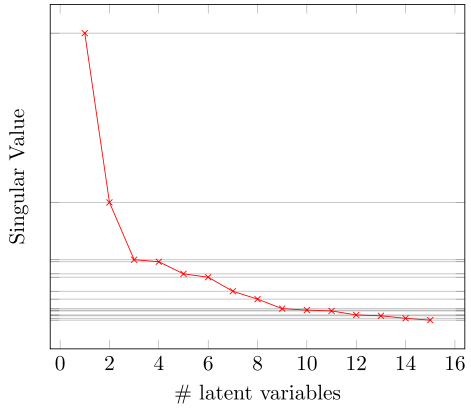

Fig. 7. Single-values varying with the number of latent variables.

As k-NN techniques require a minimum number of neighbours, a fair comparison demands to carry out the experiment by comparing only those predicted ratings for users, or items, that have such neighborhood conditions. Thus one would compute the same coverage for both techniques.

With the objective to validate the proposal, we run the experiment in another database. The goal is to demonstrate that the proposal is valid in another dataset that has distinct characteristics compared with the first.

A 10-fold cross validation is performed in each experiment step in order to obtain reliable results. The measures of accuracy used in both experiments were the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) (McLaughlin & Herlocker, 2004).

Briefly, five experiments will be carried out to evaluate the proposal of this work:

Experiment I

Evaluation the ANN and Random Forest settings using the normalization by average user rating, eight latent variables and extraction through SVD decomposition.

Experiment II

Evaluation of Naive Bayes, Random Forest and ANN using the best configuration of the previous experiment and varying the normalization, the latent variable number and the extraction technique.

Experiment III

Comparison with the best configuration of the experiment two with four classical CF approaches: user and item nearest neighbours (using the cosine and Pearson similarity), Regularized SVD, Improved Regularized SVD, SVD++ and BPMF.

Experiment IV

Evaluation varying the minimum number of neighbours to be predicted using the best setting of the second experiment.

Experiment V

Comparing the best configurations in another dataset.

4.3. Results

In this section, we discuss and present the results obtained with different settings for COFILS showing hoe to tune its parameters. In addition, we compare our proposal with the main CF techniques in the literature.

In *Experiment I*, it was found that there is a downward trend in the error when the number of neurons increase (Fig. 8) as well as when the number of decision trees increase, in the case of Random Forest (Fig. 9). We see a stabilization of the error for 8 neurons and for 50 decision trees. In both cases, an increase in the training time is correlated with an increase in the number of parameters.

In this experiment, we found that, for 8 neurons in the hidden layer, the error tends to fall, but with an increase in the learning time. We used then 10 neurons in the following experiments. The Random Forest had a sharp initial fall and then stabilized. This stabilization was for 50 decision trees.

Experiment II aimed at verifying possible configurations and ML methods whose parameters were defined in the previous experiment. Basically there were two distinct behaviors in these results. The first is when you increase the number of features, which also improves performance, and when you increase the knowledge on the data, there is an boost in the performance up to a limit (ANN and Random Forest). The other behavior is the opposite, *i.e.*,

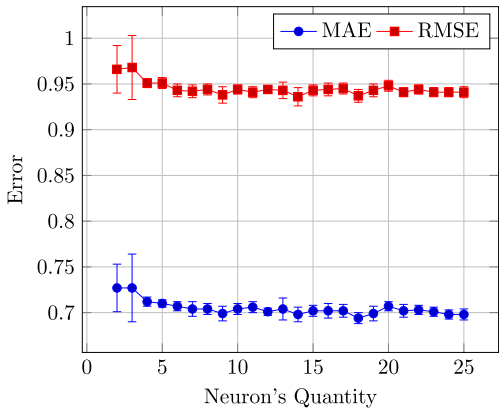

Fig. 8. Evaluation of ANN technique varying the neuron's quantity.

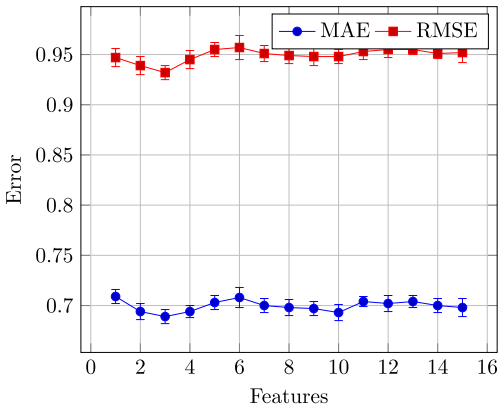

Fig. 11. Evaluation of *Experiment II* using ANN, normalization through the average user and using Partial SVD.

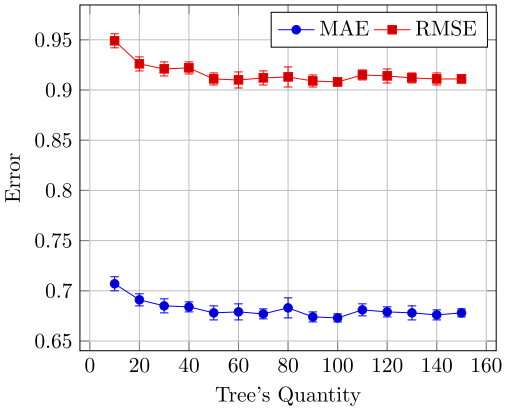

Fig. 9. Evaluation of Random Forest technique by varying the quantity of decision trees.

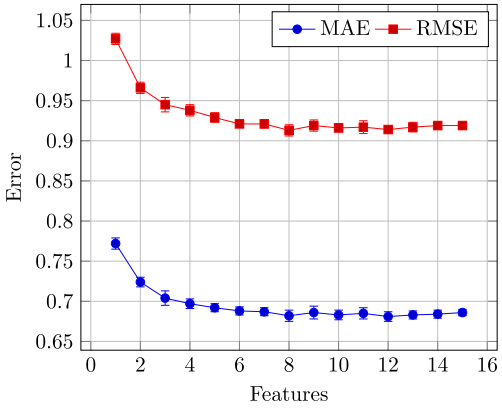

Fig. 12. Evaluation of *Experiment II* using Random Forest, normalization through the average item and using SVD.

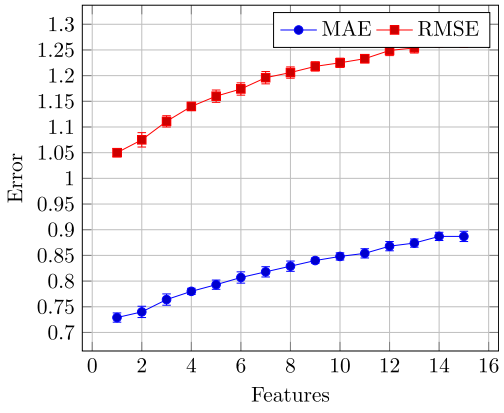

Fig. 10. Evaluation of *Experiment II* using Naive Bayes, without normalization and using Partial SVD.

worsening the performance (Naive Bayes). An example of the performance of the Naive Bayes can be seen in Fig. 10, ANN in Fig. 11 and Random Forest in Fig. 12.

Table 1 shows the best results in *Experiment II*, separating the best result for the normalization and for SL model. Our choice was to use two metrics for evaluating the performance, not necessarily the best one, thus the other two columns will show the best outcome regarding each metric presented.

As shown in the results, the representation with latent variables had a different influence on each SL model. In general, the Partial SVD generated the best result for the Naive Bayes and ANN, while pure SVD was better for Random Forest.

The amount of latent variables affects the results in different ways depending on the chosen model. The Naive Bayes got a better result with a few variables. The ANN had a better result with 4, and the Random Forest with 8 variables. Using the SVD decomposition as technique for latent variables extraction gave the best results with 8 variables. The same amount evaluated on Fig. 7.

During the proposal presentation, we hypothesized that the way to complete the missing values in the user-item matrix would directly impact the results obtained. As it can be seen from the results, the difference between using the normalization (average user rating or average item rating) against the simplest way to represent the data (using zero values) did not generate a big difference in the result, the average difference was below 1%.

Experiment III was responsible for evaluating the classical CF techniques using the same methodology that was applied to evaluate the experiments of our proposal. The Probabilistic Matrix Factorization technique obtained a better result compared to the model-based and memory-based CF techniques. But, when it is given a limit on the minimum number of neighbours for memory-based techniques, they achieve good results, even better than model-based CF techniques. On the other hand, coverage is reduced.

The results obtained using our proposed were consistently better than those obtained with the classical CF techniques. Our proposal generated an improvement of approximately 4% in MAE and 2% in RMSE, compared with the best CF techniques. These results

Table 1
Top results by varying COFILS parameter scheme (the best ones are in bold).

SL model	Normalization	Best MAE				Best RMSE			
		MAE	RMSE	Features	L.V.E.T.[a]	MAE	RMSE	Features	L.V.E.T.[a]
N. Bayes	–	0.729	1.050	1	SVDL[b]	0.729	1.050	1	SVDL[b]
ANN	–	0.691	0.930	2	SVDL[b]	0.691	0.930	2	SVDL[b]
	User	0.689	0.932	3	SVDL[b]	0.689	0.932	3	SVDL[b]
	Item	0.700	0.944	2	SVDL[b]	0.703	0.942	6	SVD
Random florest	–	0.687	0.930	6	SVDL[b]	0.690	0.918	9	SVDL[b]
	User	**0.681**	0.915	7	SVD	0.683	0.914	8	SVD
	Item	**0.681**	0.914	12	SVD	0.682	**0.913**	8	SVD

[a] Latent Variable Extraction Technique.
[b] Partial SVD by LingPipe.

Table 2
Top results on different model-based CF techniques (the best ones are in bold).

Technique	MAE	RMSE	Coverage (%)	Features
COFILS	**0.681** (0%)	0.914 (0%)	100	12
Regularized SVD	0.722 (−6.0%)	0.980 (−7.2%)	100	14
Improved Regularized SVD	0.715 (−5.0%)	0.952 (−4.2%)	100	38
SVD++	0.708 (−3.9%)	0.901 (+1.4%)	100	50
BPMF	0.698 (−2.4%)	**0.891** (+2.5%)	100	10

Table 3
Top results on memory-based CF techniques with minimum neighbors (the best ones are in bold).

Technique	MAE	RMSE	Coverage (%)	Neighbours
COFILS	0.673 (0%)	0.906 (0%)	96.40	13
CF. Item based cosine	0.723 (−7.4%)	0.963 (−6.3%)	96.42	13
CF. User based cosine	0.725 (−7.7%)	0.963 (−6.3%)	96.45	13
CF. Item based Pearson	0.713 (−5.9%)	0.954 (−5.3%)	95.01	13
CF. User based Pearson	0.715 (−5.0%)	0.955 (−5.4%)	95.05	13
CF. Item based cosine	0.700 (−4.0%)	0.936 (−3.3%)	83.14	48
CF. User based cosine	0.700 (−4.0%)	0.938 (−3.5%)	84.65	42
CF. Item based Pearson	0.683 (−1.5%)	0.922 (−1.8%)	**75.05**	49
CF. User based Pearson	0.682 (−1.4%)	0.924 (−2.0%)	**76.95**	46

can be seen in Tables 2 and 3. COFILS achieved results close to the state-of-the-art CF techniques, *i.e.*, BPMF e SVD++.

In *Experiment IV* we evaluate the performance of the best configuration obtained in *Experiment II* (Random Forest, with SVD and 8 latent variables) varying the minimum number of neighbours required to calculate the predictions. Our goal is to compare the performance of our proposal with the k-Nearest Neighbours technique. The minimum number of neighbours influences on performance when the threshold is set. Even limiting to 50, the number of neighbours in k-NN, our proposal has achieved better and more complete coverage. These result can be seen in Fig. 13 and a comparison is made with the obtained results in the Table 3.

Limiting the minimum number of neighbours in the k-NN produced a drop of about 1% MAE and 2% in RMSE. The best results occurred when we limited the number of neighbours to 13. Defining this value as a parameter, our proposal outperform k-NN of approximately 6% in MAE and 5% in RMSE.

Moreover, the best model-based techniques (Improved Regularized SVD), generated the best results when it used 38 latent variables, while COFILS already obtained an even better result with only 2 latent variables.

Only the MAE or RMSE are not enough to understand the magnitude of the recommendation being made. For this reason, the confusion matrix of the user-based CF was computed using cosine similarity and COFILS's best configuration. Table 4 is the user-based CF and Table 5 the result of COFILS.

Note that a major issue is to handle rating predictions of low and high values, since these ratings are less frequent on the dataset. In fact, recommender algorithms usually results in high error for unbalanced rating distributions. In the evaluated datasets, one should see that ratings 1, 2 and 5 are not as distributed as ratings 3 and 4, resulting in a poor prediction rate as depicted in Table 4. According to Table 4, one may note that both approaches result in low error prediction rates for ratings 3 and 4 and high error rates for ratings 1, 2 and 5. Obviously, either COFILS or memory-based CF algorithms are likely to predict better ratings 3 and 4,

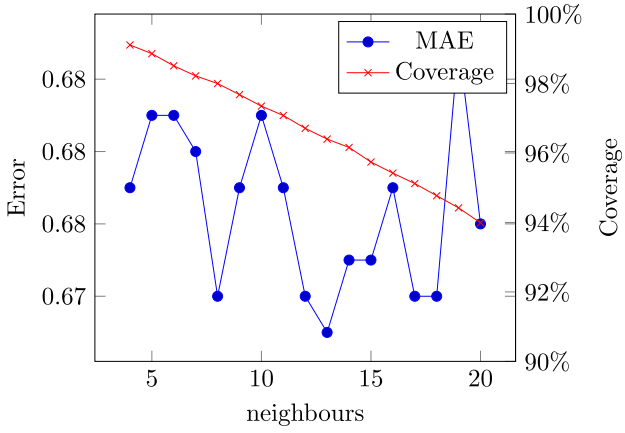

Fig. 13. Evaluation of *Experiment IV* using Random Fsorest, normalization through the average item and using SVD.

Table 4
Collaborative filtering based on user using cosine similarity and a minimum of 13 neighbours.

		Prediction				
		1	2	3	4	5
Actual	1	8.9%	33.5%	45.5%	11.8%	0.4%
	2	1.0%	18.7%	56.2%	22.9%	1.2%
	3	0.3%	7.5%	52.6%	37.9%	1.7%
	4	0.0%	2.2%	35.5%	58.1%	4.2%
	5	0.1%	1.1%	18.6%	66.6%	13.5%

Table 5
Confusion matrix of the proposed algorithm using the Random Forest, SVD, normalization by the average user and eight latent variables.

		Prediction				
		1	2	3	4	5
Actual	1	9.0%	33.1%	45.1%	12.4%	0.5%
	2	0.8%	18.6%	58.9%	21.0%	0.7%
	3	0.0%	6.8%	53.4%	39.0%	0.8%
	4	0.0%	1.4%	30.6%	63.8%	4.2%
	5	0.1%	0.8%	14.1%	67.5%	17.6%

once these are the most frequent ratings in the database. Despite this expected behavior, the proposed algorithm slightly outperforms memory-based CF technique for ratings 5.

In *Experiment V*, we conduct a similar evaluation as the previous one on a larger dataset in order to analyze the algorithm behavior. The results are provided in Table 6. Note that, again, COFILS outperforms the CF techniques showing better prediction rates. Interestingly, COFILS has improved the error rates when compared with those obtained on the smaller dataset. However, this behavior was not verified in all other CF approaches, where some of them have resulted in worse error prediction rates. This may suggest that the more instances COFILS has in the training set, the better the results are.

Overall the proposal got a better result than the k-NN predicting even more reviews. In ratings 4 and 5 there was a superior performance of the proposal compared to the classical algorithms. Another improvement was the reduction of extreme predictions, *i.e.*, when the rating is a value and the algorithm provides the worst possible score. For example, the rating is 5 and it provides 1. The only downside was the case of footnote one. The proposed algorithm gave a more extreme rating in this case than when compared with the classic one.

Through these results we can conclude that there is a considerable decrease in both the MAE as RMSE using our proposal compared with the traditional CF techniques. This comparison was made with the classical techniques within the two classes of CF: memory-based and model-based. The reduction of the MAE/RMSE was because the algorithm could better generalize the problem, as seen in the confusion matrix.

Table 6
Top CF results on 1M MovieLens dataset (the best ones are in bold).

Technique	MAE	RMSE	Coverage (%)
COFILS[a]	**0.646** (0%)	0.871 (0%)	100
COFILS[b]	0.670 (−3.7%)	0.899 (−3.2%)	100
CF. User Based Cosine[c]	0.730 (−13.0%)	0.960 (−10.2%)	99.65
CF. Item Based Cosine[c]	0.734 (−13.6%)	0.963 (−10.6%)	99.53
CF. User Based Pearson[c]	0.706 (−9.2%)	0.939 (−7.8%)	99.53
CF. Item Based Pearson[c]	0.708 (−9.6%)	0.944 (−8.4%)	99.52
Improved Regularized SVD[d]	0.690 (−6.8%)	0.915 (−5.0%)	100
SVD++[e]	0.664 (−2.9%)	0.845 (+3.0%)	100
BPMF[f]	0.659 (−2.0%)	**0.839** (+3.7%)	100

[a] Random Forest, SVD, normalization by the average user and eight latent variables.
[b] ANN, SVDL, normalization by the average user and three latent variables.
[c] 13 neighbours.
[d] 10 features.
[e] 50 features.
[f] 10 features.

5. Conclusion and outlook

In this work, we proposed a methodology to transform the CF scheme into a SL scheme through the construction of an input feature space based on the latent variables extracted from the rating matrix. Thus this methodology generates a typical supervised training set wherein instances correspond to the filled positions of the rating matrix.

Namely COFILS, the proposed methodology consists of two steps. The first step is to preprocess the rating matrix from which latent variables are extracted. These latent variables (factors) are then used in the second step that builds an input feature space to represent the ratings.

In order to evaluate the proposal, one conducted several experiments by which the obtained results are as good as the classic CF algorithms of the literature. Additionally, an empirical analysis was conducted to study how the number of factors and the method for extracting latent variables contribute to the accuracy of the rating predictions. In this analysis, Artificial Neural Networks, Naive Bayes classifiers and Random Forest classifiers were considered for comparison.

The results has proven that our method is not only comparable to the classic CF algorithms in terms of accuracy, but also reduces the rating prediction error for the classes with few observations, in case the ratings one and five. This is advantageous in RS settings, since it is important to avoid recommending those items that users definitely do not like and to recommend those items that users are likely to consume.

As future work we highlight the study of other latent variable analysis methods such as Probabilistic Latent Semantic Analysis, Tensors, and QR. We would also like to investigate the use of other supervised learning methods and the application of bootstrap to enhance rating predictions.

Another future challenge is to use the proposed methodology in a on-line recommender system scenarios, since all experiments were conducted off-line. To this end, the algorithm should be scalable and SL models should be incrementally learned by avoiding the computational cost to train the entire learning model (Huang, Zhu, & Siew, 2004, 2011; Joshi, 2012). Another future work is to apply data reduction techniques in order to avoid redundancy and noise in the training instances from which one learns a SL model (Mello, Aufaure, & Zimbrao, 2010; Sun & Wang, 2010).

Finally, the proposed method could be improved by considering other features associated with users and items such as demographic data and users' opinions and sentiments (García-Cumbreras, Montejo-Ráez, & Díaz-Galiano, 2013). These features would improve the instance representation and bring further information for the SL methods.

Acknowledgments

Our thanks to CNPq/CAPES (Brazilian Council) for funding this research.

References

Adomavicius, G., & Tuzhilin, a. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering, 17*, 734–749. http://dx.doi.org/10.1109/TKDE.2005.99.

Billsus, D. (1998). Learning collaborative information filters. *International Conference on Machine Learning*.

Candès, E. J., & Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational Mathematics, 9*, 717–772. http://dx.doi.org/10.1007/s10208-009-9045-5.

Caruana, R., & Karampatziakis, N. (2008). An empirical evaluation of supervised learning in high dimensions. In *On machine learning* (pp. 96–103). doi: 10.1145/1390156.1390169.

Cunningham, P., & Smyth, B. (2010). An assessment of machine learning techniques for review recommendation. *Artificial Intelligence and Cognitive Science*, 241–250.

Gantner, Z., Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2011). Mymedialite: A free recommender system library. In *Proceedings of the fifth ACM conference on recommender systems RecSys '11* (pp. 305–308). New York, NY, USA: ACM. doi: 10.1145/2043932.2043989.

García-Cumbreras, M. A., Montejo-Ráez, A., & Díaz-Galiano, M. C. (2013). Pessimists and optimists: Improving collaborative filtering through sentiment analysis. *Expert Systems with Applications, 40*, 6758–6765. http://dx.doi.org/10.1016/j.eswa.2013.06.049.

Hsu, S., Wen, M., Lin, H., Lee, C., & Lee, C. -h. (2007). Aimed-a personalized tv recommendation system. Interactive TV: A shared experience (pp. 166–174).

Huang, G., Zhu, Q., & Siew, C. (2004). Extreme learning machine: A new learning scheme of feedforward neural networks. *Neural Networks, 2*, 985–990. http://dx.doi.org/10.1109/IJCNN.2004.1380068.

Huang, G.-B., Wang, D. H., & Lan, Y. (2011). Extreme learning machines: A survey. *International Journal of Machine Learning and Cybernetics, 2*, 107–122. http://dx.doi.org/10.1007/s13042-011-0019-y.

Joshi, P. (2012). Incremental learning: Areas and methods – A survey. *International Journal of Data Mining & Knowledge Management Process, 2*, 43–51. http://dx.doi.org/10.5121/ijdkp.2012.2504.

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). New York, NY, USA: ACM. doi: 10.1145/1401890.1401944.

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

Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE, 7*, 76–80.

McLaughlin, M. R., & Herlocker, J. L. (2004). A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th annual international conference on research and development in information retrieval – SIGIR '04* (p. 329). doi: 10.1145/1008992.1009050.

Mello, C., Aufaure, M., & Zimbrao, G. (2010). Active learning driven by rating impact analysis. In *Proceedings of the fourth ACM conference on recommender systems* (pp. 341–344). ACM.

O'Mahony, M., & Smyth, B. (2009). Learning to recommend helpful hotel reviews. In *Proceedings of the third ACM conference on recommender systems* (pp. 305–308). ACM.

O'Mahony, M., & Smyth, B. (2010). Using readability tests to predict helpful product reviews. In *Adaptivity, personalization and fusion of heterogeneous information* (pp. 164–167). LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop* (Vol. 2007, pp. 5–8).

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). Recommender systems handbook. *Media*. http://dx.doi.org/10.1007/978-0-387-85820-3.

Salakhutdinov, R., & Mnih, A. (2008a). Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on machine learning ICML '08* (pp. 880–887). New York, NY, USA: ACM. doi: 10.1145/1390156.1390267.

Salakhutdinov, R., & Mnih, A. (2008b). Probabilistic matrix factorization. In *Advances in neural information processing systems* (Vol. 20).

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study. *Architecture*.

Schafer, J., Konstan, J., & Riedi, J. (1999). Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on electronic commerce* (pp. 158–166). ACM.

Schwartz, B. (2005). *The paradox of choice*. New York: ECCO.

Sun, L., & Wang, X. (2010). A survey on active learning strategy. *Machine Learning and Cybernetics (ICMLC)*, 11–14.