Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 417 – 426

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

Extended content-boosted matrix factorization algorithm for recommender systems

Oleksandr Krasnoshchok*, Yngve Lamo

Bergen University College, Nygrdsgaten 112, Bergen 5020, Norway

Abstract

Recommender technologies have been developed to give helpful predictions for decision making under uncertainty. An extensive amount of research has been done to increase the quality of such predictions, currently the methods based on matrix factorization are recognized as one of the most efficient.

The focus of this paper is to extend a matrix factorization algorithm with content awareness to increase prediction accuracy. A recommender system prototype based on the resulting Extended Content-Boosted Matrix Factorization Algorithm is designed, developed and evaluated. The algorithm has been evaluated by empirical evaluation, which starts with creating of an experimental design, then conducting off-line empirical tests with accuracy measurement.

The result revealed further potential of the content awareness in matrix factorization methods, which has not been fully realized in the generalized alignment-biased algorithm by Nguyen and Zhu and uncovers opportunities for future research.

Peer-review under responsibility of KES International.

Keywords: Recommender System, Matrix Factorization, Content-Boosted Matrix Factorization, Extended Content-Boosted Matrix Factorization, Gradient Descent Algorithm, Content Aware Collaborative Filtering Algorithm;

Foreword

The algorithm presented in this paper has been developed as part of the master thesis project "Recommender Systems based on Content-Boosted Matrix Factorization utilizing Ontologies"[1] written with support and in collaboration between Bergen University College, University of Bergen, Christian Michelsen Institute and company Fludo AS. The paper is organized in the following sections: introduction and motivation, related work, the algorithm description, its evaluation and results, conclusion and suggestions for future work.

* Corresponding author.
E-mail address: alex@recommender.no

Peer-review under responsibility of KES International.
doi:10.1016/j.procs.2014.08.122

Nomenclature

RS	Recommender System	ECB	Extended Content-Boosted
CF	Collaborative Filtering	TR	Traditional Matrix Factorization
CB	Content-based Filtering	MAE	Mean Absolute Error
MF	Matrix Factorization	SVD	Singular Value Decomposition
CBMF	Content-Boosted Matrix Factorization		
gAB	Generalized alignment-biased		

1. Introduction and motivation

Recommender systems (RS) are software tools and techniques providing suggestions for items to be of particular interest to a user. The suggestions relate to various decision-making processes, such as which items to buy, which music to listen to, or which online news to read[2]. Recommender systems have been an area of intensive research since 1992 and has been successfully applied in many practical applications within different domains[3,4]. Examples of such domains are movies (Netflix.com)[5], music (Pandora.com)[6], books (Goodreads.com)[7] and e-commerce (Amazon.com)[8].

RSs can be divided into three principal types: Collaborative Filtering (CF), Content-based Filtering (CB) and Hybrid. *Collaborative filtering* systems analyze people's behavior to create recommendations; *content-based* systems use only semantic information from the domain of recommendation, and *hybrid* systems are merely a combination of these two approaches. If one compares hybrid systems with collaborative or content-based systems, the recommendation accuracy is usually higher in hybrid systems. The reason is a lack of information about domain dependencies in collaborative filtering, and about people's preferences in content-based systems. The combination of them leads to a common knowledge increase, which contributes to better recommendations[9]. The knowledge increase makes it especially promising to explore new ways to extend the underlying collaborative filtering algorithms with content data and content-based algorithms with user behavior data.

Underlying recommender algorithms are a subject of continual research by various teams[10,11]. Many of the most efficient algorithms for RSs are based on the matrix factorization (MF) technique. Matrix factorization is a mathematical technique used to split (factorize) the original matrix into a product of matrices, which when multiplied return the original matrix. Algorithms based on this technique have proved their efficiency in a three-year-long competition organized by Netflix 2006 - 2009[12]. After the Netflix competition the matrix factorization method has been recognized as one of the most efficient collaborative filtering algorithms. The algorithm is based on the state of the art of recommender systems.

Improvement over traditional (with traditional we mean the baseline method as described in[13]) plain matrix factorization method for RSs is an up-to-date topic.[14,15] are examples of recent publications aimed to improve the traditional algorithm. Content awareness is one of the principal disadvantages and challenges of the CF type algorithms, namely, they use only people's behavior to produce recommendations and are not aware of the predicted content's metadata.

The challenge to introduce the content awareness in MF algorithms has been addressed in many articles such as[13,16,17]. A remarkable solution to the content awareness problem has been suggested by Nguyen and Zhu in[13], they present several content-based matrix factorization methods, which inject metadata awareness in the MF method. This innovation has shown an improvement over the traditional MF and revealed the potential for further research.

Several factors have contributed heavily to our motivation to seek further prediction quality improvements utilizing content awareness in the matrix factorization method. These are the success of recommender systems, content-based features analysis techniques such as[18] and the work of Nguyen and Zhu[13]. In this paper we will further develop the content aware approach.

2. Related work

In the past years several improvements to the matrix factorization approach have been developed. The improvements have been done in various areas such as performance increase in the Divide-and-Conquer MF[19], accuracy

increase in the Non-negative MF[16], incorporating of content information into the matrix factorization methods as in Content-Boosted MF[13], among others.

The content-boosted matrix factorization generalized alignment-biased algorithm (gAB) is one of several content-boosted algorithms developed by Jennifer Nguyen and Mu Zhu at University of Waterloo, Canada. Although the traditional matrix factorization method is not content aware, content awareness is a popular topic for research. One remarkable work, similar to[13], was made by Manzato et al. in[17]. In this work the gSVD++ algorithm has been presented. The gSVD++ algorithm utilizes content information in the matrix factorization algorithm by introducing latent feature vectors of attributes.

Another notable approach is presented in[20]. The approach supports the importance of the content information in the utilization of ontological profiling within the recommender systems domain, the paper was written in 2004 by Stuart E. Middleton. The work introduces user profiles utilizing ontological reasoning of information received from content-based filter. To produce recommendations, the recommender system compared the user profiles with items' content information described by an ontology.

The work[21] suggests incorporation of the social influence, namely, the similarity of a person to his or her friends in a social network, into the matrix factorization method. The procedure is at some degree similar to the one described in[13] and to the one used in our algorithm, with the principal difference that the penalty aimed to user latent feature vectors is applied on item latent feature vectors.

Unlike hybrid systems, in[13] it is introduced a set of content-based matrix factorization (CBMF) methods injecting metadata awareness directly into the MF algorithm. The notable difference between them is that metadata is inserted in the algorithm's procedure itself instead of making post-procedure corrections to calculated results. Our Extended Content-Boosted Matrix Factorization algorithm (ECB) is based on the techniques developed in[13] and extends it with an extra penalty, which will be described in the next section.

3. Algorithm

Our algorithm is a based on the Content-Boosted Matrix Factorization technique introduced by Nguyen and Zhu in[13]. Content-Boosted Matrix Factorization is based on the traditional matrix factorization method. In this section we start with description of the traditional matrix factorization method, and show how it is extended to CBMF. Finally, we present our new Extended Content-Boosted Matrix Factorization Algorithm.

3.1. Matrix factorization

Assume, $U = \{u_1, ..., u_n\}$ is a set of users and $I = \{i_1, ..., i_m\}$ is a set of items. Information about users and items is represented in the matrix $\mathbf{R} = U \times I$ of size $N \times M$, the entries of which are ratings r_{ui} given by user u to item i.

The set of known entries in the matrix $\mathbf{R}$ is denoted as $T = \{(u, i) : r_{ui}\ is\ known\}$. The goal of RS is to predict the unknown entries out of the set T, namely, to predict the entries where $(u, i) \notin T$. These unknown entries are denoted as $\hat{r}_{ui}$. Finally, $T_{u\bullet}$ is a set of items rated by user u, and $T_{\bullet i}$ is a set of users rated item i.

We will now recall the traditional matrix factorization. To predict missing ratings in $\mathbf{R}$, $\mathbf{R}$ is approximated by the product of two low-rank approximated[22] matrices $\mathbf{P}$ of size $N \times K$ and $\mathbf{Q}$ of size $M \times K$, which when multiplied return the best approximation of the original matrix $\mathbf{R}$:

$$\mathbf{R} \approx \hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^T = \underbrace{\begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_N^T \end{bmatrix}}_{N \times K} \underbrace{[\,\mathbf{q}_1\ \mathbf{q}_2\ \cdots\ \mathbf{q}_M\,]}_{K \times M}, \tag{1}$$

where $\mathbf{p}_u$ ($u = 1, 2, ..., N$) is a *K-dimensional latent feature* vector of user u, and $\mathbf{q}_i$ ($i = 1, 2, ..., M$) is a *K-dimensional latent feature* vector of item i. Each K-th value of $\mathbf{p}_u$ represents a preference of user u, and each K-th value of $\mathbf{q}_i$ represents a degree on which item i supports this preference.

In practice those K-dimensional vectors in the multidimensional latent space represent users' preferences and items' aspects. Preferences might be interests of users, while aspects are different properties of items such as a movie genre, length, quality, among others. Vectors, close to each other in the multidimensional space, represent similar preferences or aspects. For example, vectors of movies Terminator 2 and Terminator 3 are expected to be close to each other, because both movies possess similar aspects.

Similarly to [13], this concept forms the basis of our ECB algorithm, namely, if two items have one or several attributes in common, then their feature vectors should be close to each other in the latent space. The prediction of the rating $\hat{r}_{ui}$ given by user u to item i is merely: $\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$.

The optimization problem, which the MF method solves in practice, is:

$$\min_{\mathbf{P},\mathbf{Q}} \|\mathbf{R} - \mathbf{P}\mathbf{Q}^T\|^2, \tag{2}$$

where $\|\cdot\|$ is the Frobenius norm. The solution for this problem is to find two K-rank matrices $\mathbf{P}$ and $\mathbf{Q}$, which when multiplied return the matrix as close as possible to the original $\mathbf{R}$. The effect from the overfitting problem [23] is reduced by adding a regularization penalty λ applied to $\mathbf{P}$ and $\mathbf{Q}$:

$$\min_{\mathbf{P},\mathbf{Q}} L_{TR}(\mathbf{P},\mathbf{Q}) = \min_{\mathbf{P},\mathbf{Q}} \|\mathbf{R} - \mathbf{P}\mathbf{Q}^T\|^2 + \lambda(\|\mathbf{P}\|^2 + \|\mathbf{Q}\|^2) \tag{3}$$

Considering that the MF method trains its model only over known ratings in $\mathbf{R}$ and that we can evaluate (3) only for the known entries, the actual minimization problem becomes:

$$\min_{\mathbf{P},\mathbf{Q}} L_{TR}(\mathbf{P},\mathbf{Q}) = \textstyle\sum_{(u,i)\in T}(r_{u,i} - \mathbf{p}_u^T\mathbf{q}_i)^2 + \lambda\left(\sum_u \|\mathbf{p}_u\|^2 + \sum_i \|\mathbf{q}_i\|^2\right) \tag{4}$$

Because most of the entries are unknown, the evaluation is done only over known entries, while the local minimum for the training set is expected to be the global minimum for the rest of the data. The subscript TR stands for the traditional matrix factorization method. Formally, the optimization problem consists of an amount of all differences between predicted and known ratings for all known ratings, and the penalized sum of all user feature vectors and all item feature vectors in matrices $\mathbf{P}$ and $\mathbf{Q}$ respectively. This forms the principal concept of the traditional matrix factorization method.

3.2. Content-boosted matrix factorization

In this subsection we describe two improvements over the traditional matrix factorization made by Nguyen and Zhu in [13], namely, a relative scaling of the penalty terms and content awareness.

The principal idea of the relative scaling of the penalty terms is that the number of users can differ significantly from the number of items. For instance, the sum of $\mathbf{p}_u$ in the second part of (4) can become much larger than the sum of $\mathbf{q}_i$, meaning that the second part of (4) will be mostly denoted by user vectors $\mathbf{p}_u$. Nguyen and Zhu in [13] have found it beneficial to scale the second penalty term on the size of $\mathbf{q}_i$, such that the penalty on $\mathbf{q}_i$ is on the same order of magnitude as on $\mathbf{p}_u$, with the factor γ. γ is defined as a coefficient of the number of users to the number of items. This turns (4) into:

$$\min_{\mathbf{P},\mathbf{Q}} L_{TR'}(\mathbf{P},\mathbf{Q}) = \textstyle\sum_{(u,i)\in T}(r_{u,i} - \mathbf{p}_u^T\mathbf{q}_i)^2 + \lambda\left(\sum_u \|\mathbf{p}_u\|^2 + \gamma\sum_i \|\mathbf{q}_i\|^2\right) \tag{5}$$

Content awareness is another improvement over the traditional matrix factorization. Assume that each item i possesses D attributes. Then vector $\mathbf{a}_i = [a_{i1}, \ldots, a_{iD}]$ represents a set of D attributes of item i. $a_{id} = 1$ means that item i possesses attribute d and $a_{id} = 0$ means it does not. Assembling M items together results in the matrix $\mathbf{A} = [a_{i,d}]_{MxD}$ describing which attributes each item has. $\mathbf{A}$ will be used to calculate similarities between the items.

The content-boosted matrix factorization generalized alignment based method (subscript gAB)[13] solves the optimization problem:

$$\min_{\mathbf{P},\mathbf{Q}} L_{gAB}(\mathbf{P},\mathbf{Q}) = L_{TR'}(\mathbf{P},\mathbf{Q}) - \underbrace{\lambda\gamma \sum_{i=1}^{M}\sum_{i'=1}^{M} w(i,i')\mathbf{q}_i^T\mathbf{q}_{i'}}_{gen.\ alignment\ penalty}, \tag{6}$$

where $w(i,i')$ is a similarity coefficient between item i and item i', and $L_{TR'}(\mathbf{P},\mathbf{Q})$ is the optimization problem (5). The similarity coefficient $w(i,i')$ is calculated for each item i as:

$$w(i,i') \propto \frac{\exp[\theta(\mathbf{a}_i^T\mathbf{a}_{i'} - c)]}{1+\exp[\theta(\mathbf{a}_i^T\mathbf{a}_{i'} - c)]}, \tag{7}$$

where θ is a number selected individually from $-\infty$ to ∞. Suggested values are 0.5, 1 and 1.5[13]. Depending on the number of shared attributes c, it slightly change the similarity coefficient value range. The total sum of coefficients $w(i,i')$ is normalized to 1 for each item i.

The optimization problem (5) is extended to (6) with *generalized alignment penalty*, which penalizes item feature vectors by some small amounts "pushing" them towards each other in the latent space. This effect is called the Differential Shrinkage effect in[13] and forms the basis for the generalized alignment-biased algorithm (gAB).

3.3. Extended content-boosted matrix factorization

In this subsection we evolve the CBMF gAB algorithm further to our Extended Content-Boosted Matrix Factorization algorithm.

Recall that in the MF method the closeness of two vectors is defined as their inner dot product, therefore, the word "close" means that their inner product is larger; or inner product of items i and i', namely, $\mathbf{q}_i^T\mathbf{q}_i'$ is larger. While doing research about the MF method and particularly the CBMF gAB algorithm, we found beneficial to update latent feature vectors of all items similar to the current item with an extra penalty. Each time when $w(i,i')\mathbf{q}_i^T\mathbf{q}_{i'}$ is summed up on the right side of (6), the vector $\mathbf{q}_{i'}$ is updated as follows:

$$\mathbf{q}_{i'} = \mathbf{q}_{i'} + \underbrace{\mathbf{q}_{i'}w(i',i)\lambda\eta\mu)}_{extra\ penalty}, \tag{8}$$

with the scaling factor μ:

$$\mu_{j+1} = \mu_j v, \tag{9}$$

where v is the decaying parameter, η is the learning rate and λ is the regularization penalty. v is defined with some particular value such as 1 and gradually reduces by, for instance, 0.01 each time the Gradient Descent algorithm described below iterates. The size of μ should be inversely proportional to the increase of the amount of the extra penalty in (8). μ is initialized as 1 and decreased with each iteration.

The pseudo-code of the algorithm is presented in Algorithm 1. The principal idea is that in each iteration of the matrix factorization method every item i' similar to item i should be additionally penalized with the extra penalty so their inner dot product $\mathbf{q}_i^T\mathbf{q}_{i'}$ increases and items become closer in the latent space. The penalization is content aware, its amount depends on the degree of similarity between two items defined with the similarity coefficient (7). The decaying parameter makes a central contribution to the regularization of the scaling factor keeping it proportional to increase of the extra penalty amount. Equation (6) is refined to:

$$\min_{\mathbf{P},\mathbf{Q}} L_{ECB}(\mathbf{P},\mathbf{Q}) = L_{TR'}(\mathbf{P},\mathbf{Q}) - \underbrace{\lambda\gamma \sum_{i'=1}^{M}\sum_{i=1}^{M} w(i,i')(\mathbf{q}_{i'} + \mathbf{q}_{i'}w(i',i)\lambda\eta\mu)\mathbf{q}_i^T}_{extended\ content-boosted\ penalty}, \tag{10}$$

where the subscript "ECB" stands for "Extended Content-Boosted" Matrix Factorization algorithm.

The problem (10) can be solved utilizing the Gradient Descent method[24]. It solves the problem by moving along the gradient with respect to $\mathbf{p}_u$, while keeping $\mathbf{q}_i$ fixed, and vice versa.

Gradient Descent is a local search method for minimization of a function. It achieves the local minima on a training dataset. The local minima might be a good solution for the global minimum also. For example, finding the local minima over the known set of information, namely, known ratings, shall help predict a global set of information, namely, unknown ratings.

The algorithm is initialized with two matrices $\mathbf{P}^{(0)}$ and $\mathbf{Q}^{(0)}$ populated with usually small random values of $\mathbf{p}_u$ and $\mathbf{q}_i$, and iteratively updated for all items and users in the representation matrix, namely, for all $u = 1, \ldots, N$ and all $i = 1, \ldots, M$. The updating equations are:

$$\mathbf{p}_u{}^{(j+1)} = \mathbf{p}_u^{(j)} - \eta \nabla_u^{TR'}(\mathbf{p}_u^{(j)}, \mathbf{q}_i^{(j)}), \tag{11}$$

$$\mathbf{q_i}^{(j+i)} = \mathbf{q}_i^{(j)} - \eta \nabla_i^{ECB}(\mathbf{p}_u^{(j)}, \mathbf{q}_i^{(j)}), \tag{12}$$

where j is the number of iteration, η is a learning rate, sometimes called a step size of the gradient, and the derivatives for users and items are, correspondingly:

$$\nabla_u^{TR'} \propto \textstyle\sum_{i \in Tu.} -(r_{ui} - \mathbf{p}_u^T \mathbf{q}_i)\mathbf{q}_i + \lambda \mathbf{p}_u, \tag{13}$$

$$\nabla_i^{ECB} \propto \textstyle\sum_{u \in T.i} -(r_{ui} - \mathbf{p}_u^T q_i)\mathbf{p}_u + \lambda\gamma \left[\mathbf{q}_i - \sum_{i'=1}^{M} w(i,i')(\mathbf{q}_{i'} + \mathbf{q}_{i'} w(i',i)\lambda\eta\mu)\right] \tag{14}$$

When the improvement in the $j+1$ iteration becomes less than some threshold ε, the algorithm stops. Its output is two K-rank lower-dimensional matrices, which when multiplied together return an approximation of the original matrix. The ratings for all missing values are merely $\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$.

Now, the ECB MF algorithm can be illustrated by the following pseudo-code:

Algorithm 1 ECB MF algorithm pseudo-code.

Input: $\mathbf{R} = [r_{ui}]_{N \times M}$, $\mathbf{K}$
Output: P, Q
1: **initialize** j ← 0 and choose $\mathbf{P}^{(0)}$, $\mathbf{Q}^{(0)}$
2: **repeat**
3: **for all** $u = 1, ..., N$ and $i = 1, ..., M$ **do**
4: compute $\nabla_u^{TR'}$ and ∇_i^{ECB} with (13), (14)
5: update $\mathbf{p}_u^{j+1}$ and $\mathbf{q}_i^{j+1}$ with (11), (12)
6: **end for**
7: **until** $[L_{ECB}, (\mathbf{P}^{(j)}, \mathbf{Q}^{(j)}) - L_{ECB}(\mathbf{P}^{(j+1)}, \mathbf{Q}^{(j+1)}]/L_{ECB}(\mathbf{P}^{(j)}, \mathbf{Q}^{(j)} < \varepsilon$
8: **return** P, Q

To summarize, the algorithm starts with the original matrix $\mathbf{R}$ and the number K of dimensions with $\mathbf{P}^{(0)}$ and $\mathbf{Q}^{(0)}$ as initial input matrices populated with small random values, j is the number of iterations starting at first with 0; the algorithm iterates over all known ratings for each user and each item, until $\mathbf{P}$ and $\mathbf{Q}$ approximate the original matrix $\mathbf{R}$ such that the improvement of the minimization problem (namely, the difference between $\mathbf{PQ}^T$ and $\mathbf{R}$) for the current iteration $j+1$ compared with the previous iteration j becomes less than the threshold value ε.

The evaluation, key algorithm's parameters and results are discussed in the next sections.

4. Evaluation

Evaluation of recommender systems can be done utilizing several principal approaches, namely, off-line experiments, user studies and online experiments[25]. *Off-line experiments* are performed on pre-collected datasets of users' choices or ratings of items. *User studies* are conducted with a group of people by giving them a set of test tasks and recording observed behavior, collecting quantitative measurements, results of questionnaires, and so forth. *Online*

experiments provide the strongest evidence that the recommender system has a value. This type of experiment is usually conducted after the off-line and user studies have been done, and the system is ready to be used in production.

The importance of proper evaluation is obvious and it might be beneficial to consider several aspects when evaluating recommender systems. Both accuracy and non-accuracy points of view are important. Depending on the evaluation type, various aspects can be evaluated. Online and user study evaluations are powerful strategies enabling most non-functional and functional measurements, however off-line analysis is commonly used and requires fewer resources, but the evaluation is limited to accuracy measurements.

To evaluate the algorithm, we have utilized an off-line experimental approach with focus on the algorithm's prediction accuracy. Online and user study experiments are addressed in the future work section of this paper. To measure accuracy in off-line experiments, a dataset containing relevant test data must been selected, then it is randomly split into two parts: first as a training set and the second as a test set of ratings. The algorithm learns from the training set and makes assumptions about ratings in the test set.

The difference between predicted and real ratings forms a basis for an accuracy metrics. One of the widely used accuracy metrics is the Mean Absolute Error[25]. It measures overall error differences between a predicted rating and the real rating to the total number of ratings in the test set. The Mean Absolute Error is calculated as:

$$MAE = \sqrt{\frac{1}{|T|}\sum_{(u,i)\epsilon T}|\hat{r}_{ui} - r_{ui}|}, \tag{15}$$

where T is the test set, $\hat{r}_{ui}$ is the predicted rating for item i given by user u and r_{ui} is the real rating in the test set.

We have built a RS prototype to perform experiments and measure the prediction accuracy with MAE.

4.1. Experiment conditions

The ECB MF algorithm has been compared with the traditional matrix factorization and the gAB content-boosted matrix factorization algorithms.

The matrix factorization methods have a set of parameters influencing their performance and accuracy, namely, the regularization penalty, the scaling factor, the learning rate, among others. To have the initial conditions as close as possible to those in[13], we attempted to utilize values from[13] whenever it was suitable. To keep the conditions persistent, we kept the same parameters for each experiment and each algorithm, namely, our ECB MF, the traditional MF and gAB CBMF.

4.2. Dataset

Similarly to[13] the MovieLens 100K dataset[26] has been selected. It has 100, 000 ratings provided by 943 users on 1682 movies. The rating scheme is 1 to 5 and each user has rated at least 20 movies. Most movies are assigned to one or several genres. The sparse ratio for the dataset is 0.0636 meaning that 6.36% of all possible user-item ratings are known. 19 genres are those attributes utilized to calculate the similarity coefficient (7).

4.3. Initialization strategy

The initialization of input matrices $\mathbf{P}^{(0)}$ and $\mathbf{Q}^{(0)}$ is important. Same as in[13] we have employed a *mixed initialization* strategy utilizing the Singular Value Decomposition (SVD) technique[27].

The idea is to apply SVD to the representation matrix $\mathbf{R}$ keeping unknown entries being zero. The SVD decomposes the matrix $\mathbf{R}$ as:

$$\mathbf{R} \approx \mathbf{P}\mathbf{D}\mathbf{Q}^T \tag{16}$$

As the result, the initial $\mathbf{P}^{(0)}$ matrix becomes:

$$\mathbf{P}^{(0)}_{SVD} = \mathbf{P}\mathbf{D}^{1/2} \tag{17}$$

$\mathbf{D}^{1/2}$ is a square root of the diagonal matrix $\mathbf{D}$. The initial $\mathbf{Q}^{(0)}$ matrix becomes:

$$\mathbf{Q}^{(0)}_{SVD} = \mathbf{Q}\mathbf{D}^{1/2} \tag{18}$$

One more step to the final versions of $\mathbf{P}^{(0)}$ and $\mathbf{Q}^{(0)}$ is to add some degree of randomness to the SVD initialization. The matrix $\mathbf{P}^{(0)}$ was initialized as:

$$\mathbf{P^{(0)}} = k\mathbf{P}^{(0)}_{SVD} + (1-k)\mathbf{P}^{(0)}_{RANDOM}, \tag{19}$$

where entries in $\mathbf{P}_{RANDOM}{}^{(0)}$ are given by the Gaussian distribution, namely, $N(0, \sigma^2)$, the mean of distribution denoted as 0 and deviation σ^2. After a series of experiments we have chosen $\sigma = 0.7$ as the most efficient.

$k = 0$ means that SVD initialization is not applied as can be seen from (19). When $k = 1$ the SVD results are utilized. Other values between 0 and 1 introduce a degree of randomness to initial $\mathbf{P}^{(0)}$ and $\mathbf{Q}^{(0)}$ matrices. $k = 0.5$ is still introducing randomness and allowing the MF algorithms to learn from the model utilizing the training set.

By replacing $\mathbf{P}$ with $\mathbf{Q}$ in (19), the same initialization strategy was applied to $\mathbf{Q}^{(0)}$.

4.4. Selection of ECB parameters v and μ

The idea of the scaling factor μ is to regularize the amount of an additional penalty on sizes of item latent feature vectors. The penalty is applied by the ECB algorithm on every iteration and its size depends on the number of similar items. An item's latent feature vector will receive a larger penalty the more similar items the item has.

For example, if the item has $1,000$ similar items, it will receive $1,000$ penalties, this is not a problem when the penalty amount is small. Those $1,000$ small penalties would make two items close in the latent space. However, if each penalty from a single similar item is, for instance, equal to 1, the item's latent feature vector would receive a penalty of a size $1,000$, which might be too big value for the Gradient Descent algorithm and it will fail.

The parameter v of the scaling factor μ ensures that on every iteration the amount of a new penalty is reduced by some factor, preventing its uncontrolled increase in nearly to the geometrical progression. If $\mu = 1$, the item's latent feature vector receives the full amount of the penalty from all similar items, and if $\mu = 0$, the amount of extra penalty is also 0. This can be seen from (9).

The chosen value of $v = 0.99$ for the MovieLens 100K dataset and $\mu = 1$ were merely a best guess based on a series of test runs. Additional research should be done to evaluate how the change of the parameter v affects the resulting prediction accuracy.

4.5. Parameters summary

All the required parameters are summarized in Table 1. These are the convergence value ε, the number of dimensions K, the regularization factor λ, the scaling factor γ ensuring a balance between penalties applied to user and item latent features vectors (N is the number of users and M is the number of items), the learning rate η, coefficients θ and c, and ECB parameters v and μ.

Each line in Table 1 corresponds to one algorithm and its test conditions, crosses marked with "−" means that the parameter is not applicable for specific algorithm.

Table 1. Parameters summary

	ε	K	λ	γ	η	θ	c	k	σ	v	μ
TR	0.005	5	0.075	–	0.006	–	–	0.5	–	–	–
gAB	0.005	5	0.075	N/M	0.006	1	1	0.5	–	–	–
ECB	0.005	5	0.075	N/M	0.006	1	1	0.5	0.7	0.99	1

4.6. Results

Having all the parameters and experiment conditions defined, the number of iterations has been limited to 250 and datasets were randomly split in two equally large sets. Each algorithm has been executed 20 times in a loop and the resulting Mean Absolute Error values were averaged.

When evaluating the significance of the results it is beneficial to be aware of aspects which might influence the outcome by strengthening or weakening the results. These are elements of uncertainty.

Table 2. Results

Algorithm	Mean Absolute Error	Improvement over the TR, %
TR	0.75432	0
gAB	0.75260	0, 23%
ECB	0.74646	1, 05%

One element is that people do change opinions, one day a person rates the movie with the rating "5", a week later the person might rate the same movie as "4". This means that even if the mean absolute error is low, it does not mean that the predicted rating will be the same as the person would rate this item today.

Another element of uncertainty is that the dataset was randomly split in two parts 50% each, initial $\mathbf{P}^{(0)}_{RANDOM}$ and $\mathbf{Q}^{(0)}_{RANDOM}$ were initialized with Gaussian distribution, then all the experiments were executed 20 times each and the results were averaged. Two random elements are involved. Perhaps, increasing the number of repetition to $1,000$ would give more weighted averages, but still there will be an element of uncertainty.

Additional notable observation is that the total amount of penalty on sizes of item feature vectors in Algorithm 1 increases from iteration to iteration. This tendency can be seen in equations (12) and (14), and it has been observed when we traced Algorithm 1. Our experiment has shown that controlled change in degree of increment might lead to positive changes in prediction accuracy. Namely, we have gradually reduced the amount of extra penalty from iteration to iteration with the decaying factor v in (9). Even if the improvement of our algorithm can be seen as numerically insignificant, we think the latter observation might be utilized in different implementations of the matrix factorization methods applied to the recommender systems domain.

5. Conclusion and future work

The evaluation indicates that our ECB MF algorithm may produce improved prediction accuracy compared with the CBMF gAB and the traditional MF algorithms. Despite the improvement is relatively small, we think there is a further potential of incorporating content awareness in MF methods, which uncovers opportunities for further research.

An additional contribution is that controlled change in degree of increment might lead to positive changes in prediction accuracy. This property might be applied to a variety of modified matrix factorization methods employed in the recommender system domain.

As of today (May 2014) the ECB algorithm is integrated, but not activated in a recommender system used for recommendations of products at youabout.com. In July 2014 we plan to activate the algorithm and to perform user experiments to test functional and non-functional aspects. Examples of planned functional experiments are time spent in the system, while non-functional ones are users' satisfaction, usability of the system and users' trust to recommendations. These aspects do not affect the quality of recommendations, but have direct influence on satisfaction of those using the system. To participate in testing one can register at http://recommender.no[28].

For the algorithm's improvement areas, it might be beneficial to evaluate the ECB MF algorithm on the variety of other datasets rather than the one been used in this paper. It might also be beneficial to analyze how the scaling factor μ and the parameter v change the recommendation quality.

Another interesting idea is to apply the same principles as were used in the Extended Content-Boosted Matrix Factorization Algorithm to the latent user vectors instead of latent item feature vectors. To be precise, if two users are similar, it makes it intuitive to make their latent feature vectors close in the latent space. An alternative is to introduce the ECB MF algorithm's principles to both user and items feature vectors.

Ontology and the ECB MF algorithm is another interesting idea to combine, the use of ontological reasoning might increase the number of common attributes leading to improvement in the prediction accuracy.

References

1. Krasnoshchok, O.. *Recommender Systems based on Content-Boosted Matrix Factorization utilizing Ontologies*. Master's thesis; Bergen University College, University of Bergen; Norway; 2013.
2. Kantor, P.B., Ricci, F., Rokach, L., Shapira, B.. *Recommender Systems Handbook*. New York, NY, USA: Springer; 1st ed.; 2010.
3. Belkin, N.J., Croft, W.B.. Information filtering and information retrieval: Two sides of the same coin? *Commun ACM* 1992;**35**(12):29–38.

4. Foltz, P.W., Dumais, S.T.. Personalized information delivery: An analysis of information filtering methods. *Commun ACM* 1992;**35**(12):51–60.
5. Netflix, I.. Netflix movie recommender web site. `http://netflix.com`; 2014. Accessed: 2014-05-25.
6. Pandora, I.. Pandora internet radio. `http://pandora.com`; 2014. Accessed: 2014-05-25.
7. Goodreads.com, . Goodreads books recommender web site. `http://goodreads.com`; 2014. Accessed: 2014-05-25.
8. Amazon.com, . Amazon.com e-commerce web site. `http://amazon.com`; 2014. Accessed: 2014-05-25.
9. Burke, R.. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction* 2002;**12**(4):331–370.
10. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.. Grouplens: An open architecture for collaborative filtering of netnews. In: Smith, J.B., Smith, F.D., Malone, T.W., editors. *Proceedings of CSCW '94*. ACM; 1994, p. 175–186.
11. Dror, G., Koenigstein, N., Koren, Y., Weimer, M.. The yahoo! music dataset and kdd-cup'11. In: Dror, G., Koren, Y., Weimer, M., editors. *Proceedings of KDD Cup 2011 competition*; vol. 18. JMLR.org; 2012, p. 8–18.
12. Netflix, I.. The netflix prize web site. `http://www.netflixprize.com`; 2014. Accessed: 2014-03-14.
13. Nguyen, J., Zhu, M.. Content-boosted matrix factorization techniques for recommender systems. *Statistical Analysis and Data Mining* 2013;**6**(4):286–301.
14. Weimer, M., Karatzoglou, A., Bruch, M.. Maximum margin matrix factorization for code recommendation. In: Bergman, L.D., Tuzhilin, A., Burke, R.D., Felfernig, A., Schmidt-Thieme, L., editors. *Proceedings of the 3rd ACM Conference on RSs*; RecSys '09. ACM; 2009, p. 309–312.
15. Salakhutdinov, R., Mnih, A.. Bayesian probabilistic matrix factorization using markov chain monte carlo. In: Cohen, W.W., McCallum, A., Roweis, S.T., editors. *Proceedings of ICML '08*. New York, NY, USA: ACM; 2008, p. 880–887.
16. Lee, D.D., Seung, H.S.. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems* 2000; **13**:556–562.
17. Manzato, M.G.. gsvd++: Supporting implicit feedback on recommender systems with metadata awareness. In: Shin, S.Y., Maldonado, J.C., editors. *Proceedings of the 28th Annual ACM Symposium on Applied Computing*; SAC '13. New York, NY, USA: ACM; 2013, p. 908–913.
18. Ronen, R., Koenigstein, N., Ziklik, E., Nice, N.. Selecting content-based features for collaborative filtering recommenders. In: *Proceedings of the 7th ACM Conference on RSs*. New York, NY, USA: ACM; 2013, p. 407–410.
19. Mackey, L., Talwalkar, A., Jordan, M.I.. Divide-and-conquer matrix factorization. *NIPS* 2011;:1134–1142.
20. Middleton, S.E., Shadbolt, N.R., De Roure, D.C.. Ontological user profiling in recommender systems. *ACM Trans Inf Syst* 2004;**22**(1):54–88.
21. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.. Recommender systems with social regularization. In: King, I., Nejdl, W., Li, H., editors. *Proceedings of WSDM*. New York, NY, USA: ACM; 2011, p. 287–296.
22. Ye, J.. Generalized low rank approximations of matrices. *Mach Learn* 2005;**61**(1-3):167–191.
23. Hawkins, D.M.. The problem of overfitting. *Journal of Chemical Inform and Comp Sc* 2004;**44**(1):1–12.
24. Koren, Y., Bell, R., Volinsky, C.. Matrix factorization techniques for recommender systems. *Computer* 2009;**42**(8):30–37.
25. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.. Evaluating collaborative filtering recommender systems. *ACM Trans Inf Syst* 2004;**22**(1):5–53.
26. GroupLens, . Grouplens research datasets. `http://grouplens.org/datasets/movielens/`; 2014. Accessed: 2014-03-14.
27. Klema, V., Laub, A.J.. The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control* 1980;**25**(2):164–176.
28. Krasnoshchok, O.. Recommender systems in norway - non-profit community. `http://recommender.no`; 2014. Accessed: 2014-05-25.