sensors

MDPI

Article

Implicit Stochastic Gradient Descent Method for Cross-Domain Recommendation System

Nam D. Vo [1], **Minsung Hong** [2] **and Jason J. Jung** [1,*]

[1] Department of Computer Engineering, Chung-Ang University, 84 Heukseok, Seoul 156-756, Korea; vodinhnam@gmail.com
[2] Big Data Research Group, Western Norway Research Institute, Box 163, NO-6851 Sogndal, Norway; minsung.holdtime@gmail.com
* Correspondence: j2jung@gmail.com

Received: 21 March 2020; Accepted: 26 April 2020; Published: 29 April 2020

Abstract: The previous recommendation system applied the matrix factorization collaborative filtering (MFCF) technique to only single domains. Due to data sparsity, this approach has a limitation in overcoming the cold-start problem. Thus, in this study, we focus on discovering latent features from domains to understand the relationships between domains (called domain coherence). This approach uses potential knowledge of the source domain to improve the quality of the target domain recommendation. In this paper, we consider applying MFCF to multiple domains. Mainly, by adopting the implicit stochastic gradient descent algorithm to optimize the objective function for prediction, multiple matrices from different domains are consolidated inside the cross-domain recommendation system (CDRS). Additionally, we design a conceptual framework for CDRS, which applies to different industrial scenarios for recommenders across domains. Moreover, an experiment is devised to validate the proposed method. By using a real-world dataset gathered from Amazon Food and MovieLens, experimental results show that the proposed method improves 15.2% and 19.7% in terms of computation time and MSE over other methods on a utility matrix. Notably, a much lower convergence value of the loss function has been obtained from the experiment. Furthermore, a critical analysis of the obtained results shows that there is a dynamic balance between prediction accuracy and computational complexity.

Keywords: cross-domain; user rating consolidation; recommendation system; inner approximation; implicit update; convex optimization

1. Introduction

Recent achievements in the Internet and computing technologies have made it possible for organizations to collect, store, and process large amounts of data. These data contain detailed information related to the behaviors of users. Accurately, they represent the set of user evaluations for specific items. For example, Amazon (https://www.amazon.com/) collects information about the user's habits shopping-wise, or even regarding surfing on their website. Netflix (https://www.netflix.com/) also has substantial data related to the subject of movies. These data are beneficial for recommending useful decisions when supporting their clients. In this scenario, each firm designs a unique and maximally efficient system that can recommend as pleasant as possible items to its customers [1]. Nevertheless, not all users give ratings for items that they like or dislike. This limitation causes the fragmentation of the dataset obtained from the user, which is called data sparsity. In the real-life, a dataset is sparse at around 0.05% [2]. Therefore, the system can not produce useful recommendations when a new user or item has entered the system due to the insufficient previous ratings. This problem is named the cold-start [3], which is the most challenging issue for researchers to overcome. The cold-start problem is that problem wherein a system is not able to recommend items to users. Every recommender system is required to build a user's profile by considering his/her preferences and likes. The user's profile is developed by considering his/her activities and behaviors being perform with the system. Based on user's previous history and activities, the system makes decisions and recommends items consequently. Many investigations have been proposed to solve the cold-start problem by locating extra information among the intradomain objects to imply the association between a user and item [4–7]. Nevertheless, we cannot always obtain this kind of extra useful information.

On the other hand, the cold-start problem in insufficient data in one domain can be solved if another domain has relatively abundant data [8]. In other words, since there exists either implicitly or explicitly correlated between domains, we could overcome the cold-start problem by grouping multiple domains. In particular, the latent features existing among domains may improve recommendation accuracy. By this approach, a recommendation system can be built to exploit valuable information from one domain to contribute to another domain. These systems are known as cross-domain recommendation systems (CDRSs) [9]. In CDRSs, one of the most popular and efficient methods that has been used is matrix factorization collaborative filtering (MFCF) [10]. This method could handle both two major problems regarding two directions of CDRS development. The first direction focuses on collecting preference data from users and items from all domains. Oppositely, the CDRS in the second direction aims to connect domains based on other information, such as the properties of items or the social relations of users [11]. The preference data are exclusively focused on our research, since they is not affected by other information yet can be applied widely. Zhang et al. (2018) classified the preference-based CDRS into two groups: the first one is the situation in which there are no common areas between domains, while the other group has at least a partial overlap between domains [12]. In this paper, we concentrate on the first class of CDRS, where there is not any overlap among domains, since this situation is prevalent in real life. Regarding this type of CDRS, the existing method has used shared information from the items and users in domains [13]. Notably, similar information related to the item's contents and user's preferences was extracted from all domains to build the group-level knowledge, which is used for the utility matrices afterward [14]. Nevertheless, there are some limitations to this approach, since it is unsteady to transfer knowledge from one domain to the other. This unstable state will adversely affect the performance of the recommendation system [15].

Differently from previous works, to overcome the limitations mentioned above, we propose an efficient framework for a cross-domain recommendation system. In this framework, multiple domains that are presented by matrices are consolidated into one. Then we apply the MFCF to predict the unknown ratings from user to item. In this way, it is possible to extract the latent features from the user-group and item-group. An implicit update technique is adopted while optimizing the objective function to increase prediction accuracy. Additionally, the optimization convergence is significantly improved.

The main contributions of this study are as follows :

- We propose an efficient framework for a cross-domain recommendation system based on a constrained optimization model. In our model, the optimal solution and computation time are simultaneously taken into consideration.
- We devise an approximation algorithm that is suitable for objective function optimization in a cross-domain related problem. In particular, an implicit updating technique is applied to improve convergence time.
- We conduct extensive experiments on two real-world datasets to validate the effectiveness and efficiency of our method. The results demonstrate that the proposed framework can achieve better performance in comparison with the previous approach.

The remainder of the paper is organized as follows. Section 2 explains the background knowledge of MFCF in a single domain and reviews literature related to CRDS. Section 3 formally defines the problem formulation. In Section 4, we present our conceptual framework for CDRS. Section 5 presents an experiment. Finally, we draw conclusions and suggest directions for future study in Section 6.

2. Related Work and Background

2.1. Related Work

Recent researchers have studied cross-domain related work, as mentioned in [11,16,17], wherein there are two types of cross-domain recommended tasks. The first task is to use the information of the source domain to enhance the quality of the target domain recommendation [18–20]. Karatzoglou et al. used a machine learning method to transfer dense knowledge from the source domain to the target areas, which is much more sparse [21]. Enrich et al. used the user tags as connections between multiple domains, from which they learn the users' rating models to gain performance in the target domain [22]. The second task is recommending items in separate domains concurrently. They proposed a method for creating a rating matrix, which is the multidisciplinary shared latent factor [23,24]. Shi et al. [25] used the user-generated tags to calculate the similarity between cross-domain users and items, respectively, and then integrated these similarities into a matrix factorization model to improve the recommended accuracy. Gao et al. presented the clustering latent factor model based on a joint non-negative matrix framework [26].

For recommendation using matrix factorization, work was done by Gogna et al. [27]. They proposed a matrix completion framework that can be implemented in different domains. Zhenzhen et al. presented a cross-domain recommendation algorithm to overcome cold-start and sparsity problems and mentioned that this could be extended to consider temporal dynamics, as user preferences may change over time [28]. A cross-domain collaborative framework for recommending the venue proposed by Farseev et al. [29] is not able to address the cold-start problem. Loni et al. [30] presented a cross-domain factorization machine that can exploit additional knowledge from an auxiliary domain by encoding specific knowledge from a domain in terms of the real-valued feature vector.

In this study, we apply the MFCF for multiple domains using an updated technique to increase the convergence time of the objective function. Additionally, the implicit stochastic gradient descent-based algorithm is utilized to apply to the cross-domain recommendation system.

2.2. Background

In a single domain, let us suppose there are M users and N items. The relationship between the users and the items is presented by the user-item rating matrix $\mathbf{Y} \in \mathbb{R}^{M \times N}$, called *utility matrix*. Any rating r_{ij} in $\mathbf{Y}$ is subject to $r_{ij} \in \{1, 2, 3, 4, 5, ?\}$, where "?" represents missing value. To predict the missing values, users and items are clustered. The utility matrix $\mathbf{Y}$ can be factorized into two matrices $\mathbf{Y} \approx \hat{\mathbf{Y}} = \mathbf{X}\mathbf{W}^T$, where $\mathbf{X} \in \mathbb{R}^{M \times K}$ is the user-group membership matrix, and $\mathbf{W} \in \mathbb{R}^{N \times K}$ is the item-group membership matrix.

Figure 1 represents the matrix factorization, in which the full utility matrix $\mathbf{Y}$ is decomposed into two matrices $\mathbf{X}$ and $\mathbf{W}$, where K is much smaller than M, N. Each row in $\mathbf{X}$ represents a *user profile* $\mathbf{x}$, and each column in $\mathbf{W}$ denotes an *item profile* $\mathbf{w}$. On the other hand, the i-th item and the j-th user are represented by the i-th and j-th rows of the two matrices as $\mathbf{W}_{i*}$ and $\mathbf{X}_{j*}$. After matrix factorization, the users and items are mapped to a latent factor feature of a lower dimensionality K.

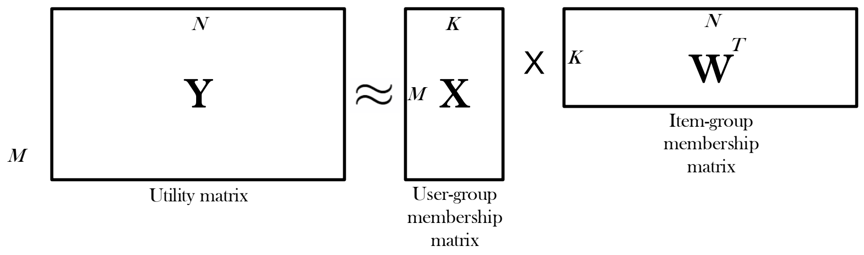

Figure 1. Decomposing utility matrix into two matrices.

To predict the missing values in the utility matrix, the low-rank matrix factorization is approximated as an optimization problem given by

$$\min_{\mathbf{X},\mathbf{W}} \mathcal{L}(f(\mathbf{X},\mathbf{W}),\mathbf{Y}) + \lambda\mathcal{R}(\mathbf{X},\mathbf{W}), \tag{1}$$

where $\mathcal{L}$ is the loss function of the predicted ratings $f(\mathbf{X},\mathbf{W})$ and the original ratings $\mathbf{Y}$, $\mathcal{R}(\mathbf{X},\mathbf{W})$ is the regularization term, and λ is the regularization tradeoff parameter. $\lambda\mathcal{R}(\mathbf{X},\mathbf{W})$ is regularization component to avoid overfitting. Regarding probabilistic matrix factorization (BMF) [31,32], the objective function to measure the loss with regularization terms and a Frobenius norm is expressed as

$$J(\mathbf{X},\mathbf{W}) = \frac{1}{2}\|\mathbf{I}\odot(\mathbf{Y}-\mathbf{X}\mathbf{W}^T)\|_F + \frac{\lambda}{2}\|\mathbf{X}\|_F + \frac{\lambda}{2}\|\mathbf{W}\|_F, \tag{2}$$

where $\mathbf{I}$ is the rating indicator matrix, $I_{ij} \in \{0,1\}$. $I_{ij} = 1$ indicates that the rating is observed, or $I_{ij} = 0$ otherwise. $\odot$ denotes the Hadamard product [33] of the matrices.

3. Problem Formulation

3.1. Definition of User-Preference Matrix

Let $\mathbf{D}_l$ with $l \in (1,L)$ be the user-preference matrix with response to l-th domain. Then, the entries of $\mathbf{D}_l$ which are denoted by $(D_l)_{ij}$ indicate the ratings of the i-th user for the j-th items of set $\mathcal{D}_l$.

By $\mathcal{U}$ we denote the set of all users that exist in multiple domains:

$$\mathcal{U} = \{\mathcal{U}^{D_1},\mathcal{U}^{D_2},\ldots,\mathcal{U}^{D_L}\}, \tag{3}$$

where $\mathcal{U}^{D_l}$ is the sets of users in l-th domain. Although these matrices are overlapping or nonoverlapping, the matrix $\mathbf{V}$, which is built from the consolidation of matrices $\mathbf{D}_1$, $\mathbf{D}_2$, ... $\mathbf{D}_L$ has the number of rows given by $|\mathcal{U}|$. Given a user $\mathrm{U}_u,\ u \in \{1,2,\ldots,|\mathcal{U}|\}$ in $\mathcal{U}$, the matrix $\mathbf{D}_l$ can be rewritten as follows:

$$\mathbf{D}_l = [(\mathbf{d}_1^{D_l})^T,\ldots,(\mathbf{d}_u^{D_l})^T,\ldots,(\mathbf{d}_{|\mathbf{U}|}^{D_l})^T]^T, \tag{4}$$

where the row vector $\mathbf{d}_u^X,\ X \in \{D_1,D_2,D_L\}$, contains the corresponding rating values of all items in $\mathcal{X} \in \{\mathcal{D}_1,\mathcal{D}_2,\mathcal{D}_L\}$, voted by user U_u. Clearly, $\mathbf{d}_u^X = \mathbf{0}$, if $\mathrm{U}_u \in \mathcal{U}\backslash\mathcal{U}^X$.

For generality, all the user's ratings can be described by the following matrix $\mathbf{V}$:

$$\mathbf{V} = [\mathbf{D}_1 \quad \mathbf{D}_2 \ldots \mathbf{D}_L]. \tag{5}$$

Matrix $\mathbf{V}$ is the expandable matrix since its dimensionality increases when adding new items and users to the data. We denote the transpose matrix of $\mathbf{V}$ by $\mathbf{V}^T$.

A column vector **b** is given as

$$\mathbf{b} = \begin{bmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \vdots \\ \mathrm{b}_n \end{bmatrix}, \tag{6}$$

where n denotes the number of rows (the number of users) of **V**. Each entry b_i is the inverse of a square root of the element a_{ii} in the $\mathbf{VV}^T$ diagonal. Therefore, b_i is as follows:

$$\mathrm{b}_i = \frac{1}{\sqrt{a_{ii}}}. \tag{7}$$

Then it is possible to write formula (6) in the following form

$$\mathbf{b} = \begin{bmatrix} \frac{1}{\sqrt{a_{11}}} \\ \frac{1}{\sqrt{a_{22}}} \\ \vdots \\ \frac{1}{\sqrt{a_{nn}}} \end{bmatrix}. \tag{8}$$

By $\mathbf{b}^T$ we denote the transform matrix of **b**, and matrix $\mathbf{B} = \mathbf{bb}^T$. In this regard, a similarity matrix **S** can be written as follows:

$$\mathbf{S} = (\mathbf{VV}^T) \odot \mathbf{B}. \tag{9}$$

This operator is a Hadamard product, in which each element p, q in **S** is the product of elements p and q of the original two matrices $(\mathbf{VV}^T)$ and **B**. After this operator, **S** will be the symmetric matrix with rows and columns being users, in which each element S_{ij} is the cosine of the angle between two vectors $\mathbf{u}_i$ and $\mathbf{u}_j$, where $\mathbf{u}_i$ and $\mathbf{u}_j$ are the i-th and j-th rows of **V**, respectively.

Remark 1. *By considering the preference vectors* $\mathbf{u}_i$ *and* $\mathbf{u}_j$*,* $i \neq j$*, the similarity between i-th and j-th users is properly given by* S_{ij}*; i.e.,*

$$S_{ij} = cos(\mathbf{u}_i, \mathbf{u}_j) = \frac{\mathbf{u}_i \mathbf{u}_j^T}{||\mathbf{u}_i||_2 . ||\mathbf{u}_j||_2}. \tag{10}$$

Given by the i-th row and j-th column entry of matrix, $\mathbf{VV}^T$ is equal to $\mathbf{u}_i\mathbf{u}_j^T$, with the corresponding entry B_{ij} of matrix **B** equivalent to

$$B_{ij} = \frac{1}{||\mathbf{u}_i||_2 . ||\mathbf{u}_j||_2}. \tag{11}$$

Therefore, (9) is considered as a generalized formulation to derive the similarity matrix among users with respect to all items.

Similarly, we can find the item-similarity matrix as follows.

$$\mathbf{K} = (\mathbf{V}^T\mathbf{V}) \odot \mathbf{C}, \tag{12}$$

where $\mathbf{C} = \mathbf{cc}^T$, **c** is a row vector $\mathbf{c} = [c_1, c_2 \ldots, c_m]$ with c_m denoting the inverse of a square root of the element d_{mm} in the $\mathbf{V}^T\mathbf{V}$ diagonal of **V**. Therefore, c_m is as follows:

$$\mathrm{c}_m = \frac{1}{\sqrt{d_{mm}}}. \tag{13}$$

Now we will factorize matrix $\mathbf{V}$. As mentioned in the previous Section, the user n gives a rating to the item m that can be approximated as $y_{mn} = \mathbf{x}_m^T \mathbf{w}_n$. However, the actual ratings have biases for users or/and items, since users tend to rate the items according to their rating behaviors, resulting in ratings that may be larger or smaller than the actual values the items receive. We use *bias* to overcome this problem. By μ_m and μ_n, we denote *biases* for *item m* and *user n*, respectively. Then the rating is approximated by

$$y_{mn} \approx \mathbf{x}_m \mathbf{w}_n + \mu_m + \mu_n + \mu, \tag{14}$$

where μ is median value of all ratings.

Therefore, the loss function (2) can be written as

$$\begin{aligned} \mathcal{L}(\mathbf{X}, \mathbf{W}, \mu_m, \mu_n) = & \frac{1}{2s} \sum_{n=1}^{N} \sum_{m=1}^{M} (\mathbf{x}_m \mathbf{w}_n + \mu_m + \mu_n + \mu - y_{mn})^2 + \\ & + \frac{\lambda}{2} (||\mathbf{X}||_F^2 + ||\mathbf{W}||_F^2 + ||\boldsymbol{\mu_m}||_F^2 + ||\boldsymbol{\mu_n}||_F^2). \end{aligned} \tag{15}$$

In the previous works, this loss function is solved by optimizing one of the pairs $(\mathbf{X}, \mu_m)$ and $(\mathbf{W}, \mu_n)$ respectively, while fixing the other pair. This process is repeated until the loss function converges. This push–pull [34] gradient method will get a sub-optimal solution [35]. In contrast with the earlier investigations, we will solve this loss function by optimizing $(\mathbf{X}, \mathbf{W}, \mu_m, \mu_n)$ simultaneously.

3.2. *Algorithm for Prediction Error Minimization*

In this section, we investigate the following joint design problem for prediction error model minimization:

$$\underset{\mathbf{X}, \mathbf{W}, \mu_m, \mu_n, \mathbf{t}}{\text{minimize}}\ \mathcal{L} = \frac{1}{2s} \sum_{n=1}^{N} \sum_{m=1}^{M} t_{mn}^2 + \frac{\lambda}{2} (||\mathbf{X}||_F^2 + ||\mathbf{W}||_F^2 + ||\boldsymbol{\mu_m}||_F^2 + ||\boldsymbol{\mu_n}||_F^2), \tag{16}$$

where $\mathbf{t} \triangleq [t_{mn}]$, $\forall m \in \{1, \dots, M\}$, $\forall n \in \{1, \dots, N\}$, with t_{mn} satisfying the following constraint:

$$\mathbf{x}_m \mathbf{w}_n + \mu_m + \mu_n + \mu - y_{mn} \le t_{mn}. \tag{17}$$

Although the objective function in (16) is a quadratic representative, which is convex, constraint (17) is still non-convex. To efficiently solve this problem, we derive a successive convex program based on an inner approximation method [36] as follows:

It is observed that (17) is equivalent to the convex constraint:

$$\sum_{k=1}^{K} u_{mnk}^2 \le t_{mn} - \mu_m - \mu_n - \mu + y_{mn} \tag{18}$$

with the following constraint imposed

$$x_{mk} w_{nk} \le u_{mnk}^2. \tag{19}$$

However, constraint (19) is still non-convex. Inspired from ([37], Lemma 1), (19) can be approximated as

$$\frac{\bar{w}_{nk}}{2\bar{x}_{mk}} x_{mk}^2 + \frac{\bar{x}_{mk}}{2\bar{w}_{nk}} w_{nk}^2 \le u_{mnk}^2, \tag{20}$$

which is convex as a second order cone constraint. Here, $\bar{x}_{mk}$ and $\bar{w}_{nk}$ are respectively the values of x_{mk} and w_{nk} at the previous iteration. Therefore, the successive convex program is formulated as

$$\underset{\mathbf{X},\mathbf{W},\mu_m,\mu_n,\mathbf{t},\mathbf{u}}{\text{minimize}} \quad \mathcal{L} = \frac{1}{2s}\sum_{n=1}^{N}\sum_{m=1}^{M} t_{mn}^2 + \frac{\lambda}{2}(||\mathbf{X}||_F^2 + ||\mathbf{W}||_F^2 + ||\boldsymbol{\mu}_{\boldsymbol{m}}||_F^2 + ||\boldsymbol{\mu}_{\boldsymbol{n}}||_F^2) \tag{21a}$$

$$\text{subject to} \quad (18), (20). \tag{21b}$$

It is realized that the problems in (21) can be efficiently solved per iteration by the existing solver (e.g., SPDT3 [38], MOSEK [39], or SeDuMi [40]), so that we obtain at least a locally optimal solution at the convergence. The algorithm for solving problem in (21) is briefly described in Algorithm 1.

Algorithm 1: Iterative algorithm for the prediction error optimization.

```
1  **Initialization:** Set $\mathcal{L}^{min} := +\infty$, $(\mathbf{x}^*, \mathbf{t}^*, w^*, u^*) := \mathbf{0}$
2  **for** each $k \in \mathcal{K}$ **do** {solving subproblem (19)}
3      **Generating an initial points:** Set $k$:= 0 and solve (20) to generate $(\mathbf{x}^{(0)}, \boldsymbol{\mu}_m^{(0)}, \mathbf{w}^{(0)}, \boldsymbol{\mu}_n^{(0)})$.
4      **repeat**
5          Solve (21) to obtain $(\mathbf{x}^*, \boldsymbol{\mu}_m^*, \mathbf{w}^*, \boldsymbol{\mu}_n^*)$ and $\mathcal{L}^{(k+1)}$.
6          Update $(\mathbf{x}^{(k+1)}, \boldsymbol{\mu}_m^{(k+1)}, \mathbf{w}^{(k+1)}, \boldsymbol{\mu}_n^{(k+1)})$:=$(\mathbf{x}^*, \boldsymbol{\mu}_m^*, \mathbf{w}^*, \boldsymbol{\mu}_n^*)$.
7          Set $k = k + 1$.
8      **until** Convergence
9      **if** $\mathcal{L}^{(k)} < \mathcal{L}^{min}$ **then**
10         Update $\mathcal{L}^{min} := \mathcal{L}^{(k)}$ and $(\mathbf{x}^*, \boldsymbol{\mu}_m^*, \mathbf{w}^*, \boldsymbol{\mu}_n^*)$:=$(\mathbf{x}^{(k)}, \boldsymbol{\mu}_m^{(k)}, \mathbf{w}^{(k)}, \boldsymbol{\mu}_n^{(k)})$.
11     **end if**
12 **end for**
```

In Algorithm 1, we use the implicit update technique to increase the convergence speed. The initial values of **x** and **w** are random. For the practical implementation, Algorithm 1 terminates upon reaching $\mathcal{L}^{(k+1)} - \mathcal{L}^{(k)} < \varepsilon$ after a finite number of iterations [41].

4. CDRS Framework

In this section, we propose a conceptual framework for a cross-domain recommender system that applies the proposed method [42]. When businesses launch multiple products or services, a mass number of data are processed to make the recommendations to clients. These data are heterogeneous and imbalanced, since their sources are from different domains. Data from users, such as ratings, number of likes, and website surfing history, are collected, clustered, and stored into the database. The cross-domain recommendation system engine will process these data to build the model. A set of parameters could be adjusted at this stage to obtain the best accuracy. The system output is the user-preferences prediction that is used to recommend items to the customers.

Particularly, according to the Figure 2, multiple datasets from various domains are preprocessed in a knowledge transfer module. Here, these data will have similarities identified and latent features extracted, and we will perform knowledge transformation. Then in the next phase, the prediction model will analyze all the information exported from the preprocessing phase in order to apply the appropriate algorithms for training and prediction generation. In this phase, most parameters are turned repeatedly to choose the best set for maximizing the whole system's accuracy. By this workflow, a CDRS can deal with heterogeneous input data and produce recommendation items in various scenarios.

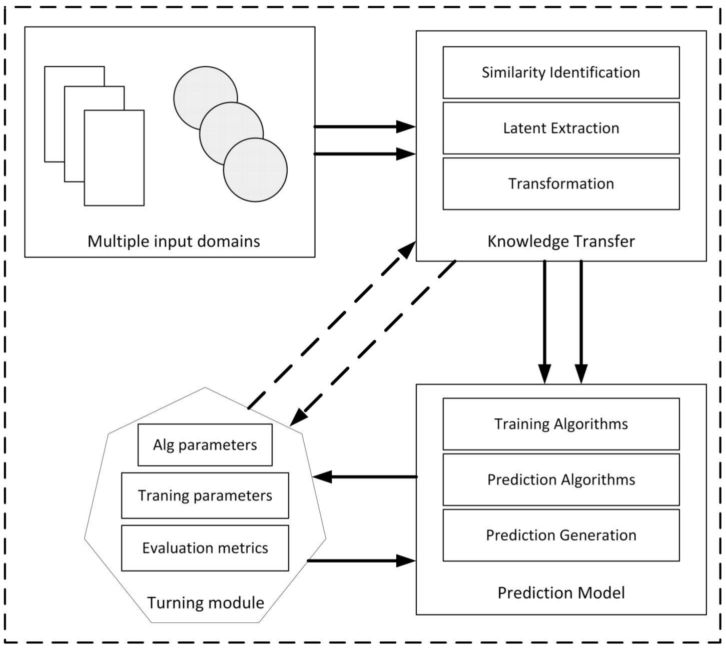

Figure 2. Conceptual framework for cross-domain recommendation system.

5. Experiments

In this section, we report experiments done to evaluate the recommendation quality of the proposed recommendation model against some baseline state-of-the-art recommendation techniques.

5.1. Dataset

To better illustrate our method, this section outlines a small-scale example. There were two datasets used: Movielens (https://movielens.org/) and Amazon Food (https://www.kaggle.com/snap/amazon-fine-food-reviews). The statistical information for these datasets is presented in Table 1.

Table 1. Statistics of datasets.

	Movielens100k	**Amazon Food**
#user	943	1072
#item	1675	1819
#rating	90,570	113,895
rating range	1–5	1–5

As shown in Table 1, the movielens100k dataset includes 943 users with 90,570 ratings for 1675 items. Therefore, its sparsity is extremely high (0.057%). Similarly, the Amazon food dataset is sparse, at around 0.058%. This sparsity is natural with respect to real-world situations in recommendation services [43]. The remaining unknown ratings are a big challenge for the recommender system to predict.

We chose three other related algorithms to compare with the proposed algorithm:

- The rating matrix generative model (RMGT): [23] one of the most popular algorithms for testing cross-domain recommended performance.
- The singular value decomposition-based MF (SVD) [44].
- The SVD++-based MF (SVD++) [45] is an extension of the SVD considering implicit ratings.

For each algorithm, we used gradient descent and implicit stochastic gradient descent, respectively, for optimization.

5.2. Evaluation Metric

We adopt the mean square error (MSE) to measure the accuracy of predicted ratings, which measures the sum of squared distances between our target ratings and predicted values. MSE is defined as follows:

$$\mathbf{MSE} = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}. \tag{22}$$

Additionally, we use mean absolute error (MAE) which has frequently been used to compare prediction errors of recommendation methods. This measurement is defined as follows:

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n}, \tag{23}$$

where n denotes the number of tested ratings, y_i is real ratings, and y_i^p is predicted ratings. This approach is used because the predicted rating values create an ordering across the items in which the predictive accuracy can also be used to measure the ability of a recommendation system to rank items with respect to user preference [46].

We use k-fold cross-validation to split the dataset. A k-fold cross-validation is where a given dataset is split into a k number of sections/folds where each fold is used as a testing set at some point. To select a proper k is important since a poorly chosen value may cause a misrepresentation of the methods. In this experiment, k is set as 10, because 5 and 10 have empirically shown to yield test error rate estimates that suffer neither from excessively high bias nor very high variance, according to [47]. Here, the dataset is split into ten folds. In the first iteration, the first fold is used to test the model, and the rest is used to train the model. In the second iteration, the second fold is used as the testing set, while the rest serves as the training set. This process is repeated until each fold of the ten folds has been used as the testing set. As we repeat the process k times, we get k times mean square error (MSE). MSE_1, MSE_2, $\ldots MSE_k$, so k-fold cross-validation error is computed by taking average of the MSE over k folds.

5.3. Baseline

A matrix factorization method is applied to solve the problem in (15). Eventually, we have to optimize the loss function $\mathcal{L}$. An optimized method based on the gradient descent algorithm is used to solve this problem. Notably, four variables will be separated into two pairs. For each iteration, one of the pairs is kept constant, while the other is optimized [48]. This process repeats sequentially until convergence is achieved based on the push–pull gradient. After convergence, the sub-optimal solution can be obtained. This solution is used as a baseline.

5.4. Experiment Parameters

Two optimization methods are used for comparison: gradient descent (GD) and implicit stochastic gradient descent (ISGD) [49,50]. The set of parameters is presented in Table 2. We have chosen these parameters based on a series of empirical tests.

Table 2. Experiment parameters.

Parameters	**Values**
Regularization parameter λ	0.01–0.1
K	10–50
Learning rate	50
Initial value of $\mathbf{w}, \mathbf{x}$	random
Number of iterations	10

5.5. Evaluation and Discussions

Now we solve the problem in this paper by optimizing all the variables simultaneously. The implicit stochastic gradient descent (ISGD) method is applied. Firstly, it is necessary to transform the original problem in (15) into the convex problem [51] formulated in (21). The parameters listed in Table 2 are the same as the baseline case. To deal with a vast quantity of variables, it is required to apply some techniques for accelerating convergence rate. Algorithm 1 shows the updating step in each iteration.

Figure 3 shows the typical convergence behavior of the algorithms for the loss function minimization problem. As a result, ISGD needs only a few iterations to reach the convergence value. Moreover, its convergence value is much lower in comparison with the baseline.

When K varies from 10 to 40, as shown in Figure 3. The slope of the ISGD convergence line also changes accordingly. When K is more extensive, this slope also increases. This leads to the initial value of the objective function also increasing significantly. The results showed the larger the K selected, the higher the objective value obtained at the first iteration. When K is larger, the dimensions of $\mathbf{x}, \mathbf{w}$ increase accordingly. This leads to an increase in the number of elements in $\mathbf{x}, \mathbf{w}$ that makes their values larger. Finally, the value of the objective function will be larger. However, the convergence value is approximately the same.

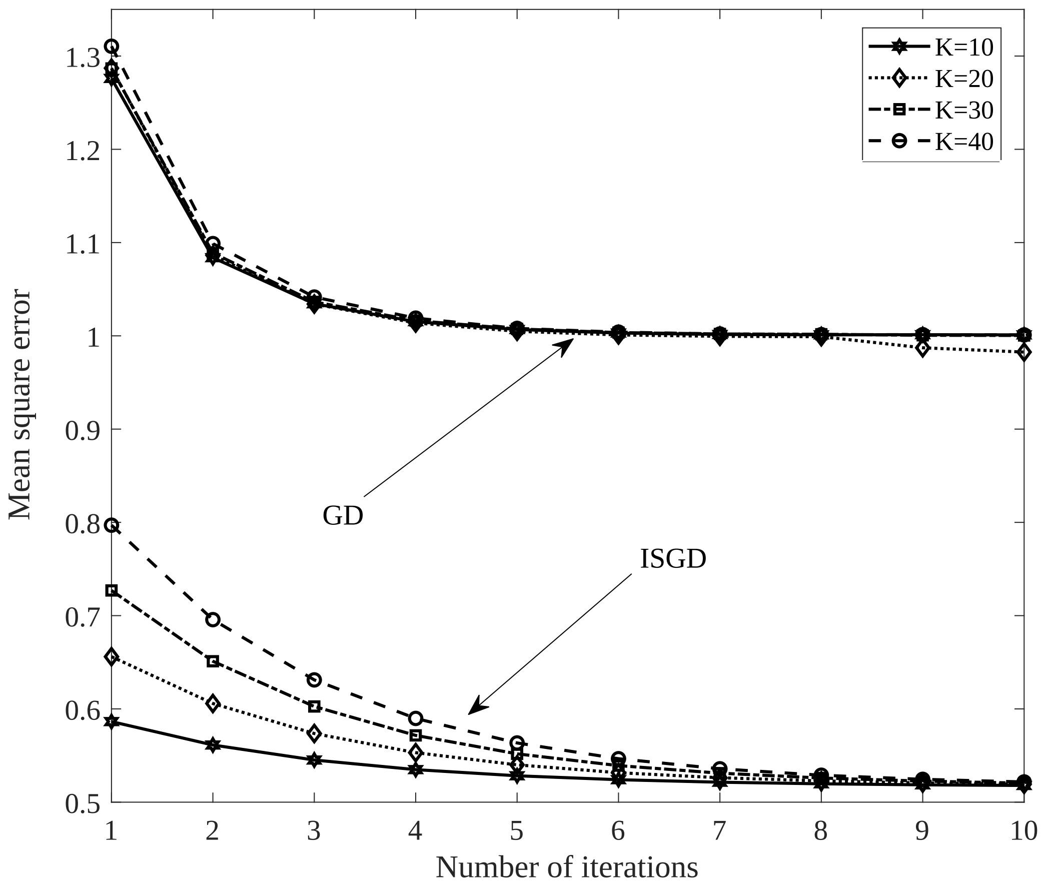

Figure 3. Typical convergence rate of GD and ISGD with varieties of K.

Let K be 10; the results according to changing the value of λ from 0.01 to 0.1 are shown in Figure 4. It shows the difference in the convergence rates when we change the regularization parameter λ. When λ is small, the objective value is obtained as a small value at the first iteration, and the convergence rate is slow. Nevertheless, with the higher λ is selected, a higher objective value is obtained at the first iteration accordingly, and the convergence value is reached faster. When K is increased (e.g., 20, 30, 40) and λ value is set as the highest value (0.1), the initial objective value is much larger since it is affected by two factors, and the convergence value is reached faster.

We recognize that the parameter K is used to adjust the approximation process. It acts the role of the dimension for approximation. The bigger K is the more accurate approximation. Nevertheless, when K increases, the value of the objective function will increase accordingly. It will be a penalty since it has a norm of **x** and **w**. This leads to a trade-off problem between the MSE and the computation complexity. K can not be so large, and the MSE has to be as small as possible.

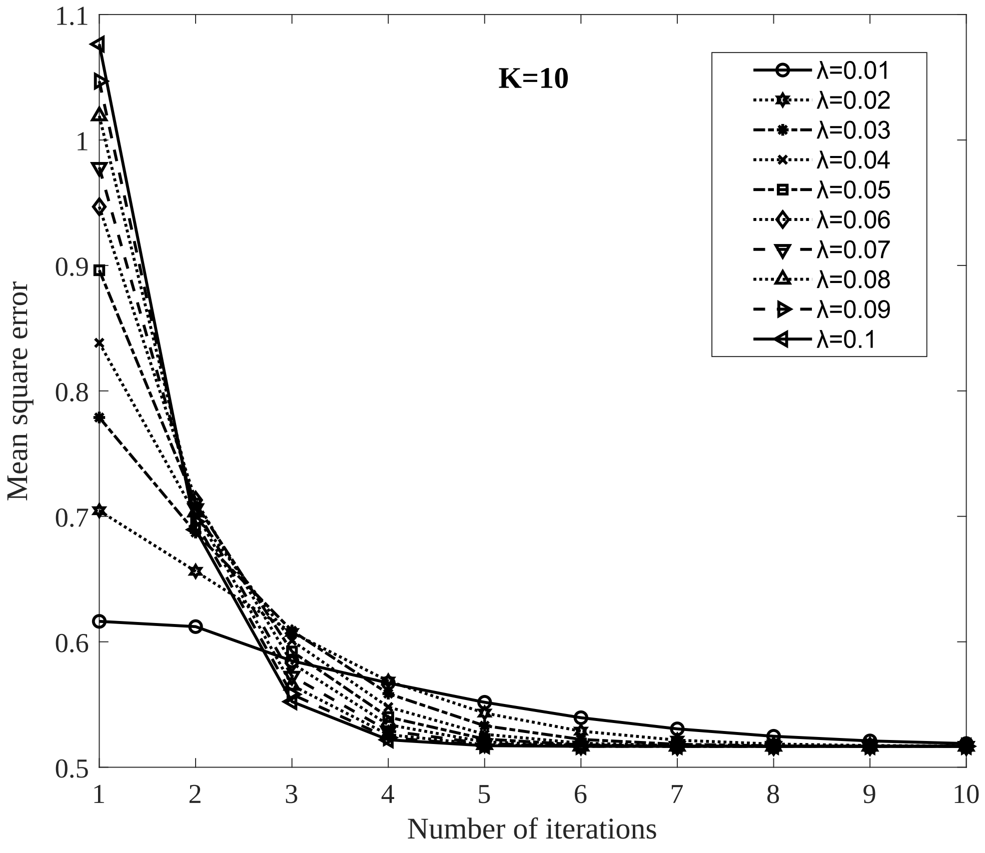

Figure 4. Typical convergence rate of ISGD with varieties of λ.

Regarding convergence time, we have measured the time until convergence between GD and ISDG. The results are shown in Table 3. In Figure 5, we can notice that the proposed method shows efficiency in terms of reducing computation time. On average, computation time has been reduced 15.2%.

Table 3. Computation time comparison (seconds).

K Value	GD Method	ISGD Method
K = 10	452	389
K = 20	583	476
K = 30	697	595
K = 40	812	699

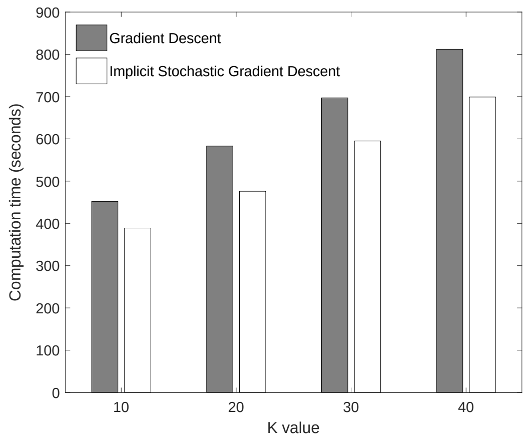

Figure 5. Computation time comparison between two methods.

Furthermore, the proposed method shows a significant result regarding prediction accuracy. Table 4 and Figure 6 show an MAE comparison between our method and other techniques. It shows that the effect of the method in this paper is better than that of other comparison methods on all tests. That is, the experimental result shows that using the ISGD technique to optimize the objective function in MFCF improves the performance of the cross-domain recommendation system.

Table 4. Comparison of MAE with other techniques.

	SVD_GD	SVD_ISGD	SVD++_GD	SVD++_ISGD	RMGM_GD	RMGM_ISGD
MAE	0.7812	0.6019	0.7964	0.5938	0.8211	0.612

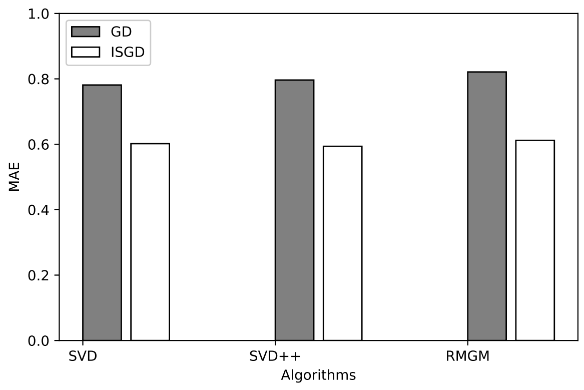

Figure 6. Comparison of MAE with other techniques.

When K varies from 10 to 40, the implicit update techniques show its efficiency to increase the convergence time. Unfortunately, the objective function has a norm of $\mathbf{x}$ and $\mathbf{w}$, which can lead to a trade-off problem between the MSE and the computation time. Additionally, our goal is to make the MSE to be as small as possible, so K can not be so large. This issue will be the limitation of our paper. We have to make a balance between the accuracy of the recommender system and the computation time.

6. Conclusions and Future Works

In this paper, we proposed a new method to consolidate multiple matrices from multiple domains for building a cross-domain recommendation system. After the consolidation, the matrix was factorized by using MFCF. The problem was to maximize the accuracy of the prediction of unknown ratings of users. To address the design problem, we transformed the original problem into sub-problems of lower dimensions. Then the iterative algorithm was proposed based on the inner approximation method to solve the sequence of convex programs. We applied the implicit stochastic gradient descent method for implicit updating each iteration. Our method with realistic parameters monotonically improved the objective function, and the convergence to a stationery point is guaranteed. Through the experiment, we demonstrated the usefulness of our approach in improving the accuracy of the CDRS.

As future work, we plan to consider using multiple data that have different distributions and attributes to test the performance of a cross-domain recommendation system. Based on this way, we can investigate the appropriate set of parameters for each specific type of data or type of domain in general.

Author Contributions: Conceptualization, N.D.V.; methodology, N.D.V.; software, N.D.V.; validation, N.D.V. and J.J.J.; formal analysis, N.D.V. and J.J.J.; investigation, N.D.V.; resources, N.D.V.; data curation, N.D.V.; writing—original draft preparation, N.D.V.; writing—review and editing, M.H. and J.J.J.; visualization, N.D.V.; supervision, J.J.J. and M.H.; project administration, J.J.J.; funding acquisition, J.J.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2019K1A3A1A80113259, NRF-2020R1A2B5B01002207).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ricci, F.; Rokach, L.; Shapira, B. Introduction to recommender systems handbook. In *Recommender Systems Handbook*; Springer: Boston, MA, USA, 2011; pp. 1–35. [CrossRef]
2. Huang, Z.; Chen, H.; Zeng, D.D. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.* **2004**, *22*, 116–142. [CrossRef]
3. Lika, B.; Kolomvatsos, K.; Hadjiefthymiades, S. Facing the cold start problem in recommender systems. *Expert Syst. Appl.* **2014**, *41*, 2065–2073. [CrossRef]
4. Li, X.; Wang, M.; Liang, T. A multi-theoretical kernel-based approach to social network-based recommendation. *Decis. Support Syst.* **2014**, *65*, 95–104. [CrossRef]
5. Li, Y.; Wu, C.; Lai, C. A social recommender mechanism for e-commerce: Combining similarity, trust, and relationship. *Decis. Support Syst.* **2013**, *55*, 740–752. [CrossRef]
6. McAuley, J.J.; Yang, A. Addressing Complex and Subjective Product-Related Queries with Customer Reviews. In Proceedings of the 25th International Conference on World Wide Web (WWW 2016), Montreal, QC, Canada, 11–15 April 2016; Bourdeau, J., Hendler, J., Nkambou, R., Horrocks, I., Zhao, B.Y., Eds.; ACM: Montreal, QC, Canada, 2016; pp. 625–635. [CrossRef]
7. Hong, M.; Jung, J.J. Multi-Sided recommendation based on social tensor factorization. *Inf. Sci.* **2018**, *447*, 140–156. [CrossRef]
8. Fernández-Tobías, I.; Cantador, I.; Kaminskas, M.; Ricci, F. Cross-domain recommender systems: A survey of the state of the art. In *Spanish Conference on Information Retrieval*; ACM: Valencia, Spain, 2012; pp. 1–12.
9. Cremonesi, P.; Tripodi, A.; Turrin, R. Cross-Domain Recommender Systems. In Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW), Vancouver, BC, Canada, 11 December 2011; Spiliopoulou, M., Wang, H., Cook, D.J., Pei, J., Wang, W., Zaïane, O.R., Wu, X., Eds.; IEEE Computer Society: Vancouver, BC, Canada, 2011; pp. 496–503. [CrossRef]
10. Koren, Y.; Bell, R.M.; Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *IEEE Comput.* **2009**, *42*, 30–37. [CrossRef]
11. Cantador, I.; Cremonesi, P. Tutorial on cross-domain recommender systems. In Proceedings of the Eighth ACM Conference on Recommender Systems (RecSys '14), Foster City, Silicon Valley, CA, USA, 6–10 October 2014; Kobsa, A., Zhou, M.X., Ester, M., Koren, Y., Eds.; ACM: Foster City, CA, USA, 2014; pp. 401–402. [CrossRef]
12. Zhang, Q.; Lu, J.; Wu, D.; Zhang, G. A Cross-Domain Recommender System With Kernel-Induced Knowledge Transfer for Overlapping Entities. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 1998–2012. [CrossRef]
13. Lu, J.; Behbood, V.; Hao, P.; Zuo, H.; Xue, S.; Zhang, G. Transfer learning using computational intelligence: A survey. *Knowl. Based Syst.* **2015**, *80*, 14–23. [CrossRef]

14. Pan, W. A survey of transfer learning for collaborative recommendation with auxiliary data. *Neurocomputing* **2016**, *177*, 447–453. [CrossRef]
15. Zhao, L.; Pan, S.J.; Yang, Q. A unified framework of active transfer learning for cross-system recommendation. *Artif. Intell.* **2017**, *245*, 38–55. [CrossRef]
16. Wang, Y.; Wang, J.; Gao, J.; Hu, S.; Sun, H.; Wang, Y. Cross-Domain Recommendation System Based on Tensor Decomposition for Cybersecurity Data Analytics. In Proceedings of the Second International Conference on Science of Cyber Security (SciSec 2019), Nanjing, China, 9–11 August 2019; Liu, F., Xu, J., Xu, S., Yung, M., Eds.; Revised Selected Papers; Springer: Nanjing, China, 2019; Volume 11933. [CrossRef]
17. Hong, M.; Akerkar, R.; Jung, J.J. Improving Explainability of Recommendation System by Multi-sided Tensor Factorization. *Cybern. Syst.* **2019**, *50*, 97–117. [CrossRef]
18. Li, B.; Yang, Q.; Xue, X. Can Movies and Books Collaborate? Cross-Domain Collaborative Filtering for Sparsity Reduction. In Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), Pasadena, CA, USA, 11–17 July 2009; Boutilier, C., Ed.; IJCAI Organization: Pasadena, CA, USA, 2009; pp. 2052–2057.
19. Kumar, A.; Kumar, N.; Hussain, M.; Chaudhury, S.; Agarwal, S. Semantic clustering-based cross-domain recommendation. In Proceedings of the 2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2014), Orlando, FL, USA, 9–12 December 2014; IEEE: Orlando, FL, USA, 2014; pp. 137–141. [CrossRef]
20. Tang, J.; Wu, S.; Sun, J.; Su, H. Cross-domain collaboration recommendation. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12), Beijing, China, 12–16 August 2012; Yang, Q., Agarwal, D., Pei, J., Eds.; ACM: Beijing, China, 2012; pp. 1285–1293. [CrossRef]
21. Karatzoglou, A.; Amatriain, X.; Baltrunas, L.; Oliver, N. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In Proceedings of the 2010 ACM Conference on Recommender Systems (RecSys 2010), Barcelona, Spain, 26–30 September 2010; Amatriain, X., Torrens, M., Resnick, P., Zanker, M., Eds.; ACM: Barcelona, Spain, 2010; pp. 79–86. [CrossRef]
22. Enrich, M.; Braunhofer, M.; Ricci, F. Cold-Start Management with Cross-Domain Collaborative Filtering and Tags. In Proceedings of the 14th International Conference on E-Commerce and Web Technologies (EC-Web 2013), Prague, Czech Republic, 27–28 August 2013; Huemer, C., Lops, P., Eds.; Springer: Prague, Czech Republic, 2013; Volume 152. [CrossRef]
23. Li, B.; Yang, Q.; Xue, X. Transfer learning for collaborative filtering via a rating-matrix generative model. In Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009), Montreal, QC, Canada, 14–18 June 2009; Danyluk, A.P., Bottou, L., Littman, M.L., Eds.; ACM: Montreal, QC, Canada, 2009; Volume 382, pp. 617–624. [CrossRef]
24. Taneja, A.; Arora, A. Cross domain recommendation using multidimensional tensor factorization. *Expert Syst. Appl.* **2018**, *92*, 304–316. [CrossRef]
25. Shi, Y.; Larson, M.A.; Hanjalic, A. Tags as Bridges between Domains: Improving Recommendation with Tag-Induced Cross-Domain Collaborative Filtering. In Proceedings of the 19th International Conference on User Modeling, Adaption and Personalization (UMAP 2011), Girona, Spain, 11–15 July 2011; Konstan, J.A., Conejo, R., Marzo, J., Oliver, N., Eds.; Springer: Girona, Spain, 2011; Volume 6787, pp. 305–316. [CrossRef]
26. Gao, S.; Luo, H.; Chen, D.; Li, S.; Gallinari, P.; Guo, J. Cross-Domain Recommendation via Cluster-Level Latent Factor Model. In Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2013), Prague, Czech Republic, 23–27 September 2013; Blockeel, H., Kersting, K., Nijssen, S., Zelezný, F., Eds.; Proceedings, Part II; Springer: Prague, Czech Republic, 2013, Volume 8189, pp. 161–176. [CrossRef]
27. Gogna, A.; Majumdar, A. Matrix completion incorporating auxiliary information for recommender system design. *Expert Syst. Appl.* **2015**, *42*, 5789–5799. [CrossRef]
28. Xu, Z.; Jiang, H.; Kong, X.; Kang, J.; Wang, W.; Xia, F. Cross-domain item recommendation based on user similarity. *Comput. Sci. Inf. Syst.* **2016**, *13*, 359–373. [CrossRef]
29. Farseev, A.; Samborskii, I.; Filchenkov, A.; Chua, T. Cross-Domain Recommendation via Clustering on Multi-Layer Graphs. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, 7–11 August 2017; Kando, N., Sakai, T., Joho, H., Li, H., de Vries, A.P., White, R.W., Eds.; ACM: Tokyo, Japan, 2017; pp. 195–204. [CrossRef]

30. Loni, B.; Shi, Y.; Larson, M.A.; Hanjalic, A. Cross-Domain Collaborative Filtering with Factorization Machines. In *Advances in Information Retrieval, Proceedings of the 36th European Conference on IR Research (ECIR 2014), Amsterdam, The Netherlands, 13–16 April 2014*; de Rijke, M., Kenter, T., de Vries, A.P., Zhai, C., de Jong, F., Radinsky, K., Hofmann, K., Eds.; Springer: Amstecdam, The Netherlands, 2014. [CrossRef]
31. Salakhutdinov, R.; Mnih, A. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 3–6 December 2007*; Platt, J.C., Koller, D., Singer, Y., Roweis, S.T., Eds.; Curran Associates, Inc.: New York, NY, USA, 2007; pp.1257–1264.
32. Ma, H.; Yang, H.; Lyu, M.R.; King, I. SoRec: Social recommendation using probabilistic matrix factorization. In Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008), Napa Valley, CA, USA, 26–30 October 2008; Shanahan, J.G., Amer-Yahia, S., Manolescu, I., Zhang, Y., Evans, D.A., Kolcz, A., Choi, K., Chowdhury, A., Eds.; ACM: Foster City, CA, USA, 2008; pp. 931–940. [CrossRef]
33. Sendov, H.S. Generalized Hadamard Product and the Derivatives of Spectral Functions. *SIAM J. Matrix Anal. Appl.* **2006**, *28*, 667–681. [CrossRef]
34. Pu, S.; Shi, W.; Xu, J.; Nedic, A. A Push-Pull Gradient Method for Distributed Optimization in Networks. In Proceedings of the 57th IEEE Conference on Decision and Control (CDC 2018), Miami, FL, USA, 17–19 December 2018; IEEE: Miami, FL, USA, 2018; pp. 3385–3390. [CrossRef]
35. Tawarmalani, M.; Sahinidis, N.V.; Sahinidis, N. *COnvexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*; Springer Science & Business Media: Heidelberg, Germany, 2002; Volume 65.
36. Marks, B.R.; Wright, G.P. Technical Note—A General Inner Approximation Algorithm for Nonconvex Mathematical Programs. *Oper. Res.* **1978**, *26*, 681–683. [CrossRef]
37. Nguyen, H.V.; Nguyen, V.; Dobre, O.A.; Nguyen, D.N.; Dutkiewicz, E.; Shin, O. Joint Power Control and User Association for NOMA-Based Full-Duplex Systems. *IEEE Trans. Commun.* **2019**, *67*, 8037–8055. [CrossRef]
38. Toh, K.C.; Todd, M.J.; Tütüncü, R.H. SDPT3—A MATLAB software package for semidefinite programming, version 1.3. *Optim. Methods Softw.* **1999**, *11*, 545–581. [CrossRef]
39. ApS, M. *The MOSEK Optimization Toolbox for MATLAB Manual*, Version 7.1 (Revision 28); MOSEK ApS: Copenhagen, Denmark, 2015; Volume 5.
40. Peaucelle, D.; Henrion, D.; Labit, Y.; Taitz, K. *User's Guide for SeDuMi Interface 1.04*; LAAS-CNRS: Toulouse, France, 2002.
41. Bestuzheva, K.; Hijazi, H. Invex optimization revisited. *J. Glob. Optim.* **2019**, *74*, 753–782. [CrossRef]
42. Vo, N.D.; Jung, J.J. Towards Scalable Recommendation Framework with Heterogeneous Data Sources: Preliminary Results. In Proceedings of the 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS 2018), Las Palmas de Gran Canaria, Spain, 26–29 November 2018; pp. 632–636. [CrossRef]
43. Singh, M. Scalability and sparsity issues in recommender datasets: A survey. *Knowl. Inf. Syst.* **2020**, *62*, 1–43. [CrossRef]
44. Webb, B. Netflix Update: Try This at Home. 2006. Available online: https://sifter.org/simon/journal/20061211.html (accessed on 20 March 2020).
45. Koren, Y. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; Li, Y., Liu, B., Sarawagi, S., Eds.; ACM: Las Vegas, NV, USA, 2008; pp. 426–434. [CrossRef]
46. Liu, X.; Liu, Y.; Aberer, K.; Miao, C. Personalized point-of-interest recommendation by mining users' preference transition. In Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM'13), San Francisco, CA, USA, 27 October–1 November 2013; He, Q., Iyengar, A., Nejdl, W., Pei, J., Rastogi, R., Eds.; ACM: San Francisco, CA, USA, 2013; pp. 733–738. [CrossRef]
47. Kuhn, M.; Johnson, K. *Applied Predictive Modeling*; Springer: Heidelberg, Germany, 2013; Volume 26.
48. Tuy, H.; Hoang, T.; Hoang, T.; Mathématicien, V.n.; Hoang, T.; Mathematician, V. *Convex Analysis and Global Optimization*; Springer: Heidelberg, Germany, 1998. [CrossRef]
49. Toulis, P.; Airoldi, E.M. Implicit stochastic gradient descent. *arXiv* **2014**, arXiv:1408.2923.

50. Yin, P.; Pham, M.; Oberman, A.M.; Osher, S.J. Stochastic Backward Euler: An Implicit Gradient Descent Algorithm for k-Means Clustering. *J. Sci. Comput.* **2018**, *77*, 1133–1146. [CrossRef]
51. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.