Deep Collaborative Filtering via Marginalized Denoising Auto-encoder

Sheng Li*
Northeastern University
Boston, MA, USA
shengli@ece.neu.edu

Jaya Kawale
Adobe Research
San Jose, CA, USA
kawale@adobe.com

Yun Fu
Northeastern University
Boston, MA, USA
yunfu@ece.neu.edu

ABSTRACT

Collaborative filtering (CF) has been widely employed within recommender systems to solve many real-world problems. Learning effective latent factors plays the most important role in collaborative filtering. Traditional CF methods based upon matrix factorization techniques learn the latent factors from the user-item ratings and suffer from the cold start problem as well as the sparsity problem. Some improved CF methods enrich the priors on the latent factors by incorporating side information as regularization. However, the learned latent factors may not be very effective due to the sparse nature of the ratings and the side information. To tackle this problem, we learn effective latent representations via deep learning. Deep learning models have emerged as very appealing in learning effective representations in many applications. In particular, we propose a general deep architecture for CF by integrating matrix factorization with deep feature learning. We provide a natural instantiations of our architecture by combining probabilistic matrix factorization with marginalized denoising stacked auto-encoders. The combined framework leads to a parsimonious fit over the latent features as indicated by its improved performance in comparison to prior state-of-art models over four large datasets for the tasks of movie/book recommendation and response prediction.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Data Mining; H.3.3 [**Information Search Retrieval**]: Information Filtering

Keywords

Collaborative filtering; matrix factorization; deep learning; denoising auto-encoder

1. INTRODUCTION

Recommendation is a fundamental problem that has gained utmost importance in the modern era of information overload. The goal of recommendation is to help users find the item that they maybe potentially interested in from a large repository of items. Recommender systems are widely used by websites (e.g., Amazon, Google News, Netflix, and Last.fm) in various contexts to target customers and provide them with useful information. A widely used setting of recommendation system is to predict how a user would rate an item (such as a movie) if only given the past rating history of the users. Many classical recommendation methods have been proposed during the last decade. The two broad categories of recommendation systems are content filtering approaches and collaborative filtering (CF) based methods. The CF based methods have attracted more attention due to their impressive performance [1]. Among various CF methods, matrix factorization has emerged as a powerful tool to perform recommendations in large datasets [2, 3].

Learning effective latent factors plays the most important role in matrix factorization based CF methods. Traditional matrix factorization methods for CF directly learn the latent factors from the user-item rating matrix [2, 4]. One of the main challenges faced by these systems is to provide a rating when a new user/item arrives in the system, which is also known as the *cold start* scenario. The cold start problem is circular in nature as - the system will not recommend an item unless it has some ratings for it and unless the system recommends it will not get ratings for it. Another practical challenge is learning the appropriate latent factors when the rating matrix is sparse, which is often the case in many real world scenarios. In order to overcome these challenges, researchers have suggested to incorporate additional sources of information about the users or items, also known as the *side information*. This side information could be obtained from the user/item profiles, for example, demographics of a user, genre of a movie, etc. The user demographics could be used to infer the relationships between the users and similarly the item similarity can be used to automatically assign ratings to new items. The use of side information to aid matrix factorization has been successfully applied by various prior work, for example [5, 6, 7]. These methods, however, only utilize the side information as regularizations in the model, and the learned latent factors may not be very effective due to the sparse nature of the ratings and the side information. In order to make matrix factorization based methods effective in such a setting, it is highly desirable to learn and extract discriminative features from the datasets.

One of the powerful approaches to capture feature interactions and complex relations that has emerged in the recent past is deep learning [8, 9]. Deep learning has attracted a lot of attention because of its promising performance to learn representations on various tasks. Deep neural networks have been shown to achieve state-of-the-art results in computer vision, speech recognition and machine translation. The application of deep learning in recommen-

*Part of the work is completed during Sheng Li's internship at Adobe Research.

CIKM'15, October 19–23, 2015, Melbourne, Australia.

DOI: http://dx.doi.org/10.1145/2806416.2806527.

dation systems, however, is very recent. With large-scale data and rich-side information available, it is now practicable to learn latent factors through deep architectures. Researchers have invested in modifying deep learning algorithms like Restricted Botzmann Machines or Convolutional Neural Networks or Deep Belief Networks directly for the task of collaborative filtering [10, 11, 12, 13, 14]. However, there are no prior work that bridge together matrix factorization with deep learning methods with the notable exception of [15]. In this paper, we present a deep learning model for collaborative filtering that tightly couples matrix factorization based collaborative filtering with deep learning algorithm namely marginalized denoising auto-encoders (mDA) [16]. Unlike [15] which integrates collaborative topic regression and bayesian stacked denoising auto-encoders and requires learning of a large number of hyper parameters using an EM style algorithm, our approach uses a much more efficient architecture based upon mDA and stochastic gradient descent and is thus computationally efficient and highly scalable. This paper makes the following contributions:

- We propose a general deep architecture named deep collaborative filtering (DCF), which integrates matrix factorization and deep feature learning. It models the mappings between the latent factors used in CF and the latent layers in deep models.
- We present a practical instantiation (i.e., mDA-CF and mSDA-CF) of the proposed architecture, by utilizing the probabilistic matrix factorization and mDA. The scalability and low computational cost of the mDA makes it a highly attractive deep learning tool, which is unlike the prior work [15].
- We evaluate the performance of our model on three real-world applications, movie recommendation, book recommendation and response prediction. Our model outperforms conventional CF methods.

2. RELATED WORK

In general, our work is closely related to the following topics: matrix factorization based collaborative filtering, and deep learning based collaborative filtering. We will discuss the two in the following subsections.

2.1 Matrix Factorization for Collaborative Filtering

The importance of accurate recommendation techniques motivated by wide ranging applications has fuelled a great amount of academic as well as industrial research in this area [17]. Recommender systems are most often based on collaborative filtering and there are typically two approaches that are widely used. In *neighborhood methods*, the similarity between users based on the content they have consumed and rated is the basis of a new recommendation. A related but intrinsically more powerful approach has been the use of *latent factor models*. Matrix factorization (MF) is the most popular technique to derive latent factor models and their success at the Netflix competition have highlighted their strength [18, 2]. For example, the given matrix $X \in \mathbb{R}^{N \times M}$ consisting of the item preferences of the users can be decomposed as a product of two low dimensional matrices U and V. The decomposition can be carried out by a variety of methods ranging from SVD based approaches [19] to the relatively new non-negative matrix factorization approach [20]. One classical MF method is probabilistic matrix factorization (PMF) [4]. The underlying assumption behind this method is that the prior probability distribution of the latent factors and the probability of the observed ratings given the latent factors follows a Gaussian distribution. Many algorithms have been developed to enhance the performance of PMF, by designing the Bayesian versions [21, 22, 23], or incorporating side information, such as social relationships [24, 5, 25].

Although promising, matrix factorization methods suffer from the problem of cold-start, i.e. what recommendations to make when a new user/item arrives in the system. Another problem often presented in many real world applications is data sparsity or reduced coverage. Incorporating side information has shown promising performance in collaborative filtering in such scenarios. Porteous *et al.* proposed a Bayesian matrix factorization (BMF) approach with side information and Dirichlet process mixtures [26]. A variational BMF method and a hierarchical BMF method that utilizes side information were also proposed in [27] and [28], respectively. Hu *et al.* proposed a cross-domain triadic factorization (CDTF) method [29], which leverages the information from other domains. The methods discussed above are proposed for addressing recommendation problems. Recently, MF based collaborative filtering is also applied to response prediction [30, 31]. The afore-mentioned approaches can alleviate the problem of cold start and data sparsity but might still suffer when the side information is sparse. Learning effective features is critical in matrix factorization. Recently, deep learning based methods have emerged as a powerful tool for learning representation and are widely used in many applications ranging from computer vision to speech recognition and machine translation. In this paper, our goal is to combine deep learning based methods with matrix factorization for collaborative filtering. In the next subsection, we survey the application of deep learning based methods for collaborative filtering.

2.2 Deep Learning for Collaborative Filtering

The application of deep learning models to the task of collaborative filtering is very new and there are not much attempts in this direction. Salakhutdinov *et al.* [10] were the first to apply deep learning to the task of collaborative filtering. They modified the restricted Boltzmann machines as a two-layer undirected graphical model consisting of binary hidden units and softmax visible units for the task of collaborative filtering. They designed an efficient learning procedure called the Contrastive Divergence (CD) to maximize an approximation to the true likelihood function. They also proposed a conditional RBM model and inference procedures. They tested the performance of the model on the Netflix dataset for movie recommendation and showed that their model performs well as compared to the baseline methods.

Truyen *et al.* [14] proposed ordinal Boltzmann machines for collaborative filtering. They studied the parameterizations for handling the ordinal nature of ratings, and presented the integration of multiple Boltzmann machines for user-based and item-based processes.

Recently, some deep learning models learn latent factors from content information such as raw features of audio or articles [32, 33]. Wang *et al.* [12] utilized deep belief nets (DBN) for music recommendation, which unifies feature extraction and recommendation of songs in a joint framework. They assumed that a user has a feature vector β_u drawn from a Gaussian prior and the songs have a feature vector x_v. They automatically learned the feature vectors of the songs using a deep belief network which is a generative probabilistic graphical model with hidden nodes and observation. It has millions of parameters to be learned from the training data. The authors used stacked layers of Restricted Boltzmann Machines for pretraining in an unsupervised fashion, and then employed the Maximum Likelihood Estimation (MLE) for supervised learning.

Oord *et al.* [13] addressed the music recommendation problem using the convolutional neural networks. They first conducted a weighted matrix factorization to handle implicit feedback and obtained latent factors for all songs. After that they used deep learning to map audio content to those latent factors. In particular, they extracted local features from audio signals and aggregated them into a bag-of-words representation. Finally, the deep convolutional network was employed to map this feature representation to the latent factors. They tested their algorithm on the Million song dataset and showed that their model improved the recommendation performance by augmenting the audio signals.

All the previously mentioned approaches mainly modify the deep learning algorithms for the task of collaborative filtering and do not directly couple matrix factorization with deep learning models. Most recently, Wang *et al.* [15] proposed a hierarchical Bayesian model called collaborative deep learning (CDL) which tightly couples stacked denoising auto-encoders (SDA) and collaborative topic regression (CTR). This work is the closest to our work but differs from ours in many significant ways as follows - (i) CDL utilized a Bayesian formulation of SDA. The generative process of CDL consists of drawing samples for CDL uses an EM-style algorithm for obtaining the MAP estimates of Bayesian SDA, and thus it has to learn a large number of parameters. Our model employs a more efficient architecture, marginalized SDA (mSDA), which computes the parameters in closed form and is thus highly efficient and scalable. (ii) CDL only extracts deep features for items, whereas our model learns deep features for both items and users.

3. PRELIMINARIES

Before we describe our general framework, we discuss the preliminaries as follows.

3.1 Matrix Factorization

Matrix Factorization (MF) is the most effective collaborative filtering approach. It allows us to discover the latent factors of user-item interactions by factorizing the interactions matrix into a joint latent space of user and item features respectively. It proceeds by decomposing the original rating matrix $R \in \mathbb{R}^{m \times n}$ consisting of ratings by m users for n items into two low-rank matrices $U \in \mathbb{R}^{m \times d}$ and $V \in \mathbb{R}^{n \times d}$ consisting of the user and item features respectively of rank d.

The system learns the latent factors by minimizing the following objective function -

$$\arg\min_{U,V} l(R, U, V) + \beta(\|U\|_{\mathrm{F}}^2 + \|V\|_{\mathrm{F}}^2), \tag{1}$$

where $l(R, U, V)$ is the loss function of predicting rating using the latent factors U and V and the last two terms are the regularizations used to avoid overfitting. $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm.

Many MF-based methods have been proposed, by designing a sophisticated loss function $l(R, U, V)$. Existing works usually pose some assumptions on the latent factors U and V in (1). Probabilistic matrix factorization (PMF) [4] provides a probabilistic foundation by assuming a probabilistic linear model with Gaussian observation noise and Gaussian priors on the latent factors.

$$p(R|U, V, \sigma^2) = \prod_{i=1}^{M} \prod_{j=1}^{N} \mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2)^{I_{ij}} \tag{2}$$

$$p(U|\sigma_u^2) = \prod_{i=1}^{M} \mathcal{N}(U_i|0, \sigma_u^2)\ ; p(V|\sigma_v^2) = \prod_{j=1}^{N} \mathcal{N}(V_j|0, \sigma_v^2). \tag{3}$$

The model is fitted by finding a MAP estimate of the parameters. Maximizing the log posterior leads to the following objective function that can be solved using stochastic gradient descent (SGD):

$$\arg\min_{U,V} E = \|R - UV^\top\|_{\mathrm{F}}^2 + \beta(\|U\|_{\mathrm{F}}^2 + \|V\|_{\mathrm{F}}^2).$$

To improve the recommendation performance of PMF, Bayesian probabilistic matrix factorization method (BPMF) [34] considers a full Bayesian treatment of the parameter space instead of a point estimate used in PMF. In the weighted matrix factorization (WMF), $l(R, U, V) = \|C \odot (R - UV^\top)\|_{\mathrm{F}}^2$, where C is the weight matrix [35]. Our model is based upon the probabilistic matrix factorization approach as it has shown to have a very good performance on several datasets and at the same time is computationally more efficient as compared to BPMF.

When side information are available, some MF methods make use of these additional information via regression to predict the ratings [28].

3.2 Marginalized Denoising Auto-encoder (mDA)

As a specific form of neural network, an autoencoder takes a given input and maps it (*encodes*) to a hidden representation via a deterministic mapping. Denoising autoencoders reconstruct the input from a corrupted version of the data with the motivation of learning a more robust mapping from the data. Various types of autoencoders have been developed in the literature and have shown promising results in several domains [36, 37]. Moreover, denoising autoencoders can be stacked to construct a deep network also known as stacked denoising autoencoder (SDA) which allows learning higher level representations [38]. Despite their state-of-the-art performance, one of the main drawbacks of SDA is the high computational cost of training, as they rely upon the iterative and numerical optimization techniques to learn a large amount of model parameters.

Marginalized denoising auto-encoder (mDA) [16] is a variant of SDA that avoids the high computational cost by marginalizing out the random feature corruption and thus has a closed-form solution to learn model parameters. Therefore, mDA is highly scalable and faster than SDA. It proceeds as follows:

Given a sample set $X = [x_1, \cdots, x_k]$, mDA considers multiple passes (e.g., c-times) of random corruptions over X to obtain $\tilde{X}$. It then reconstructs the input with a mapping W that minimizes the squared loss as follows:

$$\mathcal{L}(W) = \tfrac{1}{2ck} \sum_{j=1}^{c} \sum_{i=1}^{k} \|x_i - W\tilde{x}_{ij}\|^2, \tag{4}$$

where $\tilde{x}_{ij}$ represents the j^{th} corrupted version of the original input x_i and W represents the mapping that is expected to minimizes the loss function.

The above objective can be rewritten in the matrix form as

$$\mathcal{L}(W) = \|\bar{X} - W\widetilde{X}\|_{\mathrm{F}}^2, \tag{5}$$

where $\bar{X} = [X, \cdots, X]$ is the c-times repeated version of X, and $\tilde{X}$ is the corresponding corrupted version. This problem is similar to the ordinary least squares problem and has the analytical solution as given by $W = SQ^{-1}$, where $S = \tilde{X}\tilde{X}^T$, $Q = \bar{X}\tilde{X}^T$ When $c \to \infty$ in (4), we can derive the expectations of Q and P, and obtain the closed form solution of the mDA [16]. Further, multiple mDAs can be stacked to form a deep architecture, marginalized stacked denoising auto-encoder (mSDA). mSDA usually enhances the performance of mDA. Most recently, a nonlinear version of mDA is presented [39].

Table 1: Summary of notations.

Notation	Description
m	Number of users
n	Number of items
d	Dimension of latent factors
p	Dimension of user features
q	Dimension of item features
$R \in \mathbb{R}^{m \times n}$	Rating matrix
$U \in \mathbb{R}^{m \times d}$	Latent factors of users
$V \in \mathbb{R}^{n \times d}$	Latent factors of items
$X \in \mathbb{R}^{p \times m}$	Side information of users
$Y \in \mathbb{R}^{q \times n}$	Side information of items
$W_1 \in \mathbb{R}^{p \times p}$	Mapping function for X in auto-encoder
$P_1 \in \mathbb{R}^{p \times d}$	Projection matrix for U

4. OUR APPROACH

As we noted earlier, deep learning models have been proven to be very effective in extracting high-level representations from the raw input data in several learning tasks. The learned features represent high-level knowledge. In the collaborative filtering problem, we face a similar challenge of inferring effective latent and high-level knowledge on user preferences from the raw inputs, including the rating matrix and related features. MF based CF methods are able to capture the implicit relationship between the users and the items successfully, but they suffer from the cold start and data sparsity problems. Therefore, it is reasonable to draw strength from the deep models to assist the collaborative filtering process.

Table 1 summarizes the symbols used in our approach. Next, we describe a general framework that integrates matrix factorization and deep feature learning.

4.1 Deep Collaborative Filtering (DCF): A General Framework

In this section, we introduce the proposed deep collaborative filtering (DCF) framework which unifies the deep learning models with MF based collaborative filtering. Fig. 1 illustrates the idea of our DCF framework. DCF is a *hybrid* model, which makes use of both rating matrix and side information and bridges together matrix factorization and feature learning.

Given a user-item rating matrix R, the user side information X and the item side information Y, DCF jointly decomposes R and learns latent factors (i.e., U, V) from ratings and side information (i.e., X and Y) through the following formulation:

$$\arg\min_{U,V} \quad l(R,U,V) + \beta(\|U\|_{\mathrm{F}}^2 + \|V\|_{\mathrm{F}}^2) \\ +\gamma\mathcal{L}(X,U) + \delta\mathcal{L}(Y,V), \tag{6}$$

where β, γ and δ are the trade-off parameters.

There are two key components in the DCF framework: (i) the function $l(R,U,V)$ for decomposing the rating matrix R into the two latent matrices; (ii) the functions $\mathcal{L}(X,U)$ and $\mathcal{L}(Y,V)$ that connect the user/item contextual features with the latent factors. The first component derived through matrix factorization extracts latent knowledge from the rating matrix. The second component devised using deep learning models establishes connections of the side information with the latent factors.

4.2 DCF using PMF + mDA

A natural instantiation of DCF is combining probabilistic matrix factorization (PMF) with marginalized denoising auto-encoders (mDA). PMF is a widely applied CF approach with excellent performance, and mDA is a powerful tool in extracting high-level features from raw inputs. The combination of the two leverages their benefits for learning even richer models.

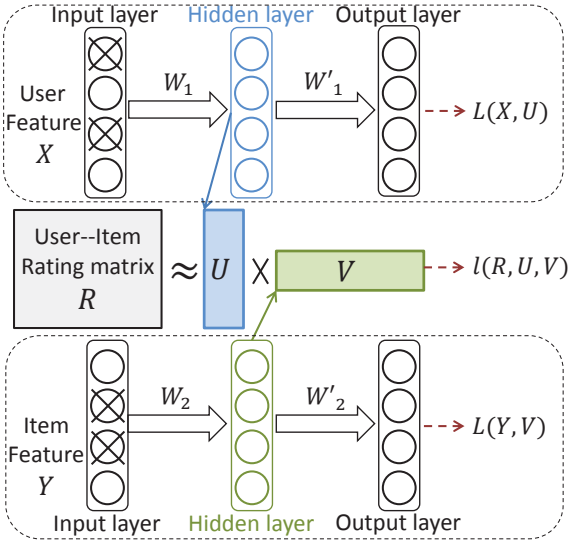

Figure 1: Illustration of DCF framework. The inputs are user-item rating matrix R, the user feature set X and the item feature set Y. Our approach jointly decomposes R and learns latent factors (i.e., U, V) from ratings and side information (i.e., X and Y). In particular, the latent factors are extracted from the hidden layer of deep networks.

4.2.1 *mDA based Collaborative Filtering (mDA-CF)*

Let $\bar{X} \in \mathbb{R}^{p \times cm}$ and $\bar{Y} \in \mathbb{R}^{q \times cn}$ denote the c-times repeated versions of X and Y respectively and let $\tilde{X}$ and $\tilde{Y}$ denote their corrupted versions. As discussed before, we utilize the loss function of PMF to decompose rating matrix R, i.e., $l(R,U,V) = \|A \odot (R - UV^\top)\|_{\mathrm{F}}^2$, where A is the indicator matrix indicating the non-empty entries in R and $\odot$ denotes the Hadamard or pointwise product. The objective function of mDA-CF is formulated as follows:

$$\arg\min_{\substack{U,V,W_1,\\ W_2,P_1,P_2}} \quad \mathcal{L}_U(W_1,P_1,U) + \mathcal{L}_V(W_2,P_2,V) + \\ \alpha\|A \odot (R - UV^\top)\|_{\mathrm{F}}^2 + \beta(\|U\|_{\mathrm{F}}^2 + \|V\|_{\mathrm{F}}^2), \tag{7}$$

where

$$\mathcal{L}_U(W_1,P_1,U) = \|\bar{X} - W_1\tilde{X}\|_{\mathrm{F}}^2 + \lambda\|P_1U^\top - W_1X\|_{\mathrm{F}}^2,$$

$$\mathcal{L}_V(W_2,P_2,V) = \|\bar{Y} - W_2\tilde{Y}\|_{\mathrm{F}}^2 + \lambda\|P_2V^\top - W_2Y\|_{\mathrm{F}}^2,$$

$W_1 \in \mathbb{R}^{p \times p}$ and $W_2 \in \mathbb{R}^{q \times q}$ are reconstructive mappings, $P_1 \in \mathbb{R}^{p \times d}$ and $P_2 \in \mathbb{R}^{q \times d}$ are projection matrices, α, β and λ are trade-off parameters. Note that, we set γ and δ in (6) to 1 for simplicity.

The first term in $\mathcal{L}_U(W_1,P_1,U)$ denotes the learning process in marginalized denoising auto-encoder. It measures the reconstruction error between input user features $\bar{X}$ and the mapped features of corrupted inputs, i.e., $W_1\tilde{X}$. W_1 is the learned mapping that is expected to minimize the loss. The second term connects the hidden layer feature W_1X and the latent factor U. Generally, the latent factor has much lower dimension than the raw features. Therefore, we add a low-dimensional projection P_1 that maps latent factor to the feature space.

Optimization

Although the optimization problem in (7) is not jointly convex in all the variables, it is convex to each of them when fixing the others. Hence, we can alternately optimize for each of the variables in (7). The detailed procedures are provided below.

First, we derive a solution to solve W_1 and W_2 using [16]. By ignoring the variables irrelevant to W_1, the objective (7) can be rewritten as

$$\arg\min_{W_1} \quad \|\bar{X} - W_1\tilde{X}\|_{\mathrm{F}}^2 + \lambda\|P_1U^\top - W_1X\|_{\mathrm{F}}^2. \tag{8}$$

Inspired by mDA, we consider the infinitely many copies of noisy data, and obtain the optimal solution

$$W_1 = E[S_1]E[Q_1]^{-1}, \tag{9}$$

where $S_1 = \bar{X}\tilde{X}^\top + \lambda P_1U^\top X^\top$ and $Q_1 = \bar{X}\tilde{X}^\top + \lambda XX^\top$. An efficient solver for calculating the expectations $E[S_1]$ and $E[Q_1]$ is provided in [16].

Similarly, we can derive the closed-form solution to W_2

$$W_2 = E[S_2]E[Q_2]^{-1}, \tag{10}$$

where $S_2 = \bar{Y}\tilde{Y}^\top + \lambda P_2V^\top Y^\top$ and $Q_2 = \bar{Y}\tilde{Y}^\top + \lambda YY^\top$.

Next, by dropping the irrelevant variables w.r.t. P_1, the objective function becomes

$$\arg\min_{P_1} \quad \lambda\|P_1U^\top - W_1X\|_{\mathrm{F}}^2. \tag{11}$$

We can obtain the closed-form solution as

$$P_1 = W_1XU(U^\top U)^{-1}. \tag{12}$$

Similarly, the optimal solution of P_2 is

$$P_2 = W_2YV(V^\top V)^{-1}. \tag{13}$$

To solve for the latent factors U and V, we use the popular stochastic gradient descent (SGD) algorithm. In particular, when other variables irrelevant to U and V are fixed, we use $f(U, V)$ to denote the objective in (7). The update rules are:

$$\begin{aligned} u_i &= u_i - \eta\tfrac{\partial}{\partial u_i}f(U,V), \\ v_j &= v_j - \eta\tfrac{\partial}{\partial v_j}f(U,V), \end{aligned} \tag{14}$$

where η is the learning rate , and the detailed derivatives are defined as

$$\begin{aligned} \tfrac{\partial f(U,V)}{\partial u_i} =& \lambda(P_1^\top(P_1u_i - (W_1X)_i)) + \beta u_i \\ & - \alpha \sum_{(i,j)\in\mathcal{A}} (R_{i,j} - u_iv_j^\top)v_j. \end{aligned} \tag{15}$$

$$\begin{aligned} \tfrac{\partial f(U,V)}{\partial v_j} =& \lambda(P_2^\top(P_2v_j - (W_2Y)_j)) + \beta v_j \\ & - \alpha \sum_{(i,j)\in\mathcal{A}} (R_{i,j} - u_iv_j^\top)u_i. \end{aligned} \tag{16}$$

The above steps are repeated until convergence. Finally, we obtain the latent factors U and V.

Algorithm Complexity

The steps of our mDA-CF approach are summarized in *Algorithm 1*. The learned latent factors U and V can be used to predict missing entries in the rating matrix. In *Algorithm 1*, we have analytical solutions of Steps 3-6 which are efficient to compute. The matrix multiplication and inversion used in Step 5 and Step 6 cost $\mathcal{O}(p^2m + pmd + d^3)$ and $\mathcal{O}(q^2n + qnd + d^3)$, respectively. The Steps 8-9 are implemented in a batch-learning fashion, and cost $\mathcal{O}(tN)$ to evaluate the gradients, where t is the number of iterations and N is the number of training ratings/responses in R. Considering that $N \gg \max\{m, n, d\}$, the time complexity of *Algorithm 1* is mainly determined by $\mathcal{O}(tN)$. Hence, our approach owns a good scalability. We discuss the settings of the parameters in the experimental section. To further reduce the computational cost, some advanced distributed optimization algorithms could be applied to our model [40].

Algorithm 1. *mDA-CF Algorithm*

Input: Rating matrix R, user features X, item features Y, parameters λ, α, β.
Output: Latent factors U, V

1: Initialize U, V, P_1 and P_2;
2: ***while*** validation error decreases, ***do***
3: Update W_1 using (9);
4: Update W_2 using (10);
5: Update P_1 using (12);
6: Update P_2 using (13);
7: ***for*** each observed R_{ij}, ***do***
8: Update u_i using (14);
9: Update v_j using (14);
10: ***end for***
11: ***end while***

4.2.2 *Stacked mDA based Collaborative Filtering (mSDA-CF)*

Existing literature show that stacking multiple deep learning layers together can usually generate rich features in the form of hidden layers, and therefore results in better performance for various learning tasks. Inspired by the marginalized stacked denoising auto-encoders (mSDA) [16], we stack multiple mDA together, and present the mSDA-CF approach.

We assume that only one hidden layer should be close to the latent factor. The reasons are two-fold. First, latent factors are high-level representations, which should correspond to the deeper layers in deep models. Secondly, latent factors should be unique, but different hidden layers have various representations. Therefore, enforcing the similarity between multiple hidden layers and latent factors is unreasonable.

In our mSDA-CF model, we assume that the latent factors are generated from the $\lfloor\frac{l+1}{2}\rfloor$ layer, given the total number of layers is l. When we train the model for the rest layers, the parameters λ, α and β are simply set to 0. In particular, if $i \neq \lfloor\frac{l+1}{2}\rfloor$, we only need to update W_1^i and W_2^i and ignore the other steps, where W_1^i and W_2^i denote the mappings in the i-th layer. One benefit of such setting is the time efficiency, as we do not increase too much computational burden when adding multiple layers.

Another interesting problem is how to set the number of layers. The number of layers implies the model complexity, which is usually related to the learning task and the size of training data. In the experiments we will discuss the influence of different number of layers. The detailed procedures of mSDA-CF are summarized in *Algorithm 2*.

4.3 Discussion

We notice that most existing deep learning based collaborative filtering methods can be unified in our DCF framework. For example, Oord *et al.* [13] use deep convolutional neural networks (CNN) to predict latent factors from music audio using the following ob-

Algorithm 2. mSDA-CF Algorithm

Input: Rating matrix R, user features X, item features Y λ, α, β, layers l.
Output: Latent factors U, V

1: ***for*** i $1 : l$, ***do***
2: ***if*** $i = \lfloor \frac{l+1}{2} \rfloor$, ***do***
3: Update U and V using *Algorithm 1*, by setting valid values to λ, α and β;
4: ***otherwise***
5: Update W_1^i and W_2^i using *Algorithm 1*, by setting $\lambda = 0$, $\alpha = 0$ and $\beta = 0$;
6: ***end if***
7: ***end for***

jective function:

$$\min_{\theta} \sum_{u,i} c_{u,i}(p_{u,i} - x_u^\top y_i')^2, \tag{17}$$

where x_u, the latent factor of user, is estimated from weighted matrix factorization beforehand, and y_i', the latent factors of audio, are learned from CNN, i.e., $y_i' = \text{CNN}(f_v, \Omega)$. Here, f_v denotes audio contents, and Ω denotes the model parameters in CNN. Eqn (17) can be interpreted in our DCF formulation (i.e., Eqn (6)). First, it utilizes the weighted matrix factorization as the loss function $l(\cdot)$. Secondly, the regularization function $\mathcal{L}(Y, V)$ is implemented by CNN.

Wang *et al.* [12] utilize deep belief networks (DBN) to build a hybrid model for collaborative filtering. The objective function is

$$\begin{aligned} L_{\text{Hybrid}} = \sum_{u,v \in I} (r_{uv} - \beta_u' x_v - r_u' y_v)^2 + \lambda_\beta \|\beta - \mu\|_{\text{F}}^2 \\ + \lambda_\gamma \|\gamma\|_{\text{F}}^2 + \lambda_y \|y\|_{\text{F}}^2. \end{aligned} \tag{18}$$

where x_v, the latent factor of item, is obtained from DBN, i.e., $x_v = \text{DBN}(f_v, \Omega)$.

Also, we can interpret model (18) in our DCF framework. First, loss function $l(\cdot)$ is implemented in a hybrid way, i.e., rating r_{uv} is predicted by the sum of the CF part $r_u' y_v$ and the content part $\beta_u' x_v$. Secondly, DBN is employed to map the content features f_v and latent factor x_v, which can be formulated by $\mathcal{L}(Y, V)$ in DCF framework. Meanwhile, γ in (6) is set to 0.

Wang *et al.* [15] propose a model with Bayesian stacked denoising auto-encoders (SDAE) and collaborative topic regression are integrated. The objective function is

$$\begin{aligned} L = & -\sum_{i,j} \tfrac{c_{ij}}{2}(r_{ij} - u_i^\top v_j)^2 - \tfrac{\lambda_n}{2} \sum_j \|f_r(X_{0,j*}, W^+) - X_{c,j*}\|_2^2 \\ & - \tfrac{\lambda_v}{2} \sum_j \|v_j - f_e(X_{0,j*}, W^+)^\top\|_2^2 - f_{reg}, \end{aligned} \tag{19}$$

where $f_e(\cdot)$ and $f_r(\cdot)$ denote the encoding and decoding functions in SDAE, λ_n and λ_v denote the trade-off parameters, and f_{reg} denote the regularization terms that prevent overfitting.

Obviously, the model (19) can also be interpreted in the DCF framework. The loss function for decomposing rating matrix in (19) is in the standard matrix factorization fashion. Further, the second and the third terms in (19) infer the item latent factor using SDAE, which can be abstracted as $\mathcal{L}(Y, V)$ in DCF.

In summary, the existing deep collaborative filtering methods [13, 12, 15] can be unified in a common framework, DCF. We also notice that the existing models only infer latent factors of items using deep models, whereas the latent factors of users are generated in traditional ways. Compared to existing works, our DCF framework provides a more flexible way to explore effective latent factors for users and/or items via deep models.

5. EXPERIMENTS

We evaluate the performance of our mDA-CF and mSDA-CF approaches on three challenging tasks that are movie recommendation, book recommendation and response prediction.

5.1 Movie Recommendation

For movie recommendation, we conduct experiments on two benchmark datasets MovieLens-100K and MovieLens-1M [1], which are commonly used for evaluating collaborative filtering algorithms.

The MovieLens-100K dataset contains 100K ratings of 943 users and 1682 movies, and the MovieLens-1M dataset consists of about 1 million ratings of 6040 users and 3706 movies. Each rating is an integer between 1 (worst) and 5 (best). The ratings are highly sparse. Table 2 summarizes the statistics of datasets. We extract the features from side information of users and movies to construct X and Y. To summarize, the user information which consists of the user's age, gender and occupation were encoded into a binary valued vector of length 28. Similarly, the item feature information which consists of the 18 category of movie genre were encoded into a binary valued vector of length 18. Ratings were normalized to be zero-mean.

As our model (7) is an extension of the representative collaborative filtering method PMF, we mainly compare our approach with PMF and its several variants, such as the Biased PMF [41] and sparse covariance matrix factorization (SCMF) [23]. PMF and Biased PMF are special cases of our approach mDA-CF. For example, by adding zero weight to the first two terms in (7), mDA-CF degrades to PMF. Further, since our approach takes advantage of the side information of users and movies, we also compare our approach with the collaborative filtering method that incorporates side information, such as the Bayesian matrix factorization with side information (BMFSI) [26].

We employ the root mean squared error (RMSE) as the evaluation metric. RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i,j} Z_{ij}^P (R_{ij} - \hat{R}_{ij})^2}, \tag{20}$$

where R_{ij} is the ground-truth rating of user i for item j, $\bar{R}_{ij}$ denotes the corresponding predicted rating, N is the total number of ratings in the test set, and Z_{ij}^P is a binary matrix that indicates test ratings.

For all the compared methods, we set the regularization parameters (e.g., λ, α and β) via 5-fold cross validation. Following the experimental settings in [23], we train each compared method with different percentages (50%, 80%, and 99%) of ratings. The training data are randomly chosen from each dataset, and the remaining data are used for testing. This process is repeated five times, and we report the average RMSE.

For the MovieLens-100K dataset, the parameters α, β and λ are set to 0.7, 0.004 and 0.2, respectively. The learning rate used in SGD is set to 0.002. Table 3 shows the average RMSE (with standard deviations) of baselines PMF, Biased PMF, BMFSI, SCMF and our approaches, mDA-CF and mSDA-CF, on the MovieLens-100K dataset. For each method, we have two settings for the dimensions of latent factors, including $d = 10$ and $d = 20$. We can observe from Table 3 that

(a) Our approaches (mDA-CF and mSDA-CF) achieve much better performance than PMF and Biased PMF, which are special cases of our approach. It demonstrates the effectiveness of incorporating side information and deep architectures.

[1] http://grouplens.org/datasets/movielens/

Table 2: Statistics of datasets used in our experiments.

Dataset	#Users	#Items	Sparsity	User Features	Item Features
MovieLens-100K	943	1682	93.7%	Age, gender, and occupation	Genres
MovieLens-1M	6040	3706	95.8%	Age, gender, and occupation	Genres
Book-Crossing	278858	271379	99.9%	Age, country, city, etc.	Title, year, publisher, etc.
Advertising	448158	737	99.7%	Age, geolocation, domain, etc.	Ad size, advertisor, etc.

Table 3: Average RMSE (with standard deviation) of compared methods with different percentages of training data on MovleLens-100K dataset.

	99%		80%		50%	
Method	d=10	d=20	d=10	d=20	d=10	d=20
PMF [4]	0.9184±0.0265	0.9164±0.0261	0.9223±0.0056	0.9190±0.0052	0.9524±0.0023	0.9506±0.0024
Biased PMF [41]	0.8953±0.0189	0.8923±0.0150	0.9135±0.0039	0.9087±0.0030	0.9388±0.0029	0.9337±0.0020
BMFSI [26]	0.8912±0.0127	0.8905±0.0154	0.9114±0.0031	0.9065±0.0029	0.9371±0.0023	0.9335±0.0025SS
SCMF [23]	0.8891±0.0146	0.8896±0.0198	0.9092±0.0033	0.9068±0.0036	0.9334±0.0025	0.9331±0.0021
mDA-CF (Ours)	0.8874±0.0142	0.8861±0.0153	0.9043±0.0043	0.9040±0.0045	0.9312±0.0026	0.9311±0.0025
mSDA-CF (Ours)	**0.8852±0.0135**	**0.8849±0.0167**	**0.9035±0.0028**	**0.9024±0.0030**	**0.9309±0.0026**	**0.9308±0.0028**

(b) BMFSI is a Bayesian matrix factorization method that utilizes side information, so it performs better than PMF and Biased PMF that ignore such information. Our approach outperforms BMFSI, which validates the strengths of the latent factors learned by marginalized denoising auto-encoders.

(c) Usually, deep models with multiple layers lead to better performance. Our mSDA-CF slightly enhances the performance of mDA-CF. We will show the influence of different number of layers in the next section.

(d) Note that the basic component in our approach is PMF. Actually, DCF is a general framework for collaborative filtering. When we implement $l(R, U, V)$ in (6) as some advanced MF methods (e.g., weighted matrix factorization), the results could be further improved.

For the MovieLens-1M dataset, the parameters α, β and λ are set to 0.8, 0.003 and 0.3, respectively. Table 4 shows the average RMSE (with standard deviations) of our approach and compared methods on the MovieLens-1M dataset. Basically, Table 4 shows similar phenomenon to that we observed from Table 3. The proposed mDA-CF and mSDA-CF approaches consistently achieve lower RMSE than compared methods. As before, we evaluate each method in two settings with different dimension of latent factors. Usually, $d = 20$ generates better results than $d = 10$ on the two MovieLens datasets.

In addition, we compare our approaches with the state-of-the-art deep learning based collaborative filtering method, a joint user-item based restricted Boltzmann machine (UI-RBM) [11]. To conduct fair comparisons with UI-RBM, we adopt the mean absolute error (MAE) used in [11] as evaluation metric. The MAE is defined as follows

$$\text{MAE} = \frac{\sum_{i}^{L} |g_i - p_i|}{L}, \tag{21}$$

where g_i is the ground truth rating, p_i is the predicted rating, and L is the total number of ratings.

We follow the experimental settings in [11], and conduct 5-fold cross-validations. The dimension of latent factors is set to 20. Table 5 shows the average MAE of compared methods on the MovieLens-100K dataset. U-RBM and I-RBM denote the user-based model and item-based model, respectively. They are the baselines used in [11]. Table 5 shows that UI-RBM achieves much better performance than traditional methods like PMF, as it takes advantage of the deep feature learning. Our mDA-CF and mSDA-CF approaches obtain lower MAE than UI-RBM, demonstrating the effectiveness of our DCF framework compared to the RBM based deep learning models.

Table 5: MAE of compared methods on MovieLens-100K dataset.

Method	MAE
PMF [4]	0.793
U-RBM [11]	0.779
I-RBM [11]	0.775
I-RBM+INB [11]	0.699
UI-RBM [11]	0.690
mDA-CF (Ours)	0.683
mSDA-CF (Ours)	**0.680**

5.2 Book Recommendation

For book recommendation, we utilize the Book-Crossing dataset [2], which contains 1149780 ratings for 271379 books from 278858 users. The rating scale is from 0 to 10 with the higher score indicating the more preference. Some attributes of users and books are also provided in this dataset. These attributes are encoded to binary vectors, which form the feature sets for users and books. Table 2 shows some statistics of this dataset.

We follow the settings in [43], and conduct 5-fold cross-validation. The baselines include PMF, the implicit social matrix factorization (ISMF) [42] and the coupled item-based matrix factorization (CIMF) [43]. ISMF incorporates the implicit social relationships

[2] http://www2.informatik.uni-freiburg.de/~cziegler/BX/

Table 4: Average RMSE (with standard deviation) of compared methods with different percentages of training data on MovleLens-1M dataset.

Method	99%		80%		50%	
	d=10	d=20	d=10	d=20	d=10	d=20
PMF [4]	0.8424±0.0071	0.8388±0.0059	0.8559±0.0022	0.8512±0.0017	0.8790±0.0009	0.8745±0.0011
Biased PMF [41]	0.8408±0.0070	0.8367±0.0067	0.8531±0.0019	0.8493±0.0020	0.8766±0.0015	0.8722±0.0012
BMFSI [26]	0.8391±0.0067	0.8340±0.0069	0.8503±0.0017	0.8478±0.0019	0.8742±0.0016	0.8703±0.0010
SCMF [23]	0.8364±0.0065	0.8323±0.0065	0.8496±0.0019	0.8465±0.0018	0.8707±0.0013	0.8678±0.0007
mDA-CF (Ours)	0.8335±0.0064	0.8317±0.0062	0.8449±0.0015	0.8429±0.0013	0.8655±0.0007	0.8645±0.0006
mSDA-CF (Ours)	**0.8320±0.0063**	**0.8304±0.0057**	**0.8416±0.0014**	**0.8407±0.0011**	**0.8628±0.0005**	**0.8613±0.0006**

Table 6: RMSE of compared methods on Book-Crossing data.

Method	RMSE ($d = 10$)	RMSE ($d = 50$)
PMF [4]	3.7483	3.7452
ISMF [42]	3.7440	3.7415
CIMF [43]	3.7398	3.7372
mDA-CF (Ours)	3.6610	3.6528
mSDA-CF (Ours)	**3.6592**	**3.6513**

between users and between items. CIMF makes use of the attributes of books in the model.

Table 6 shows the RMSE of all compared methods. We can observe that ISMF and CIMF obtain better results than the conventional PMF method, as they incorporate side information to their models. Our approaches obtain much lower RMSE than PMF, ISMF and CIMF in different settings. mSDA-CF achieves the best results among all competitors.

5.3 Response Prediction

Response prediction is another interesting application of collaborative filtering [30]. In particular, we consider a specific type of response prediction, which is click prediction. Given an online advertisement, our task is to predict whether a given user will click it or not in the near future.

Previous research works have proved that collaborative filtering (CF) methods are suitable for addressing the click prediction problem. Unfortunately, there are few datasets available for evaluating the click prediction performance of CF models. To evaluate the performance of our model in real-world applications, we collected an advertising dataset at a large software company. The dataset is collected from its website, which contains the click responses for advertisements over 3 million users. For our purpose, we analyze the data from a 2-month period, from October 1, 2013 to November 30, 2013.

The dataset used in our experiments contains 737 ads and 448,158 users. It also has the impression and click data of the advertisements. For each click event, we have the *user ID*, *day*, *ad ID*, *page ID*, *country*, *browser*, *advertiser ID*, and *size of ad*. Each record can be uniquely identified by a *(user, ad, day)* triplet. In addition, we have information about the user profiles. For each user, we have some attributes such as country, domain, etc. Apart from the user information, our dataset contains the meta-information of the ads such as ad size. In the experiments, we encode the demographic information (e.g., country, state, domain) into a binary valued vector for each user. The attributes of ads (e.g., advertiser, ad size) are also encoded into binary vectors. Some statistics of this dataset can be found in Table 2.

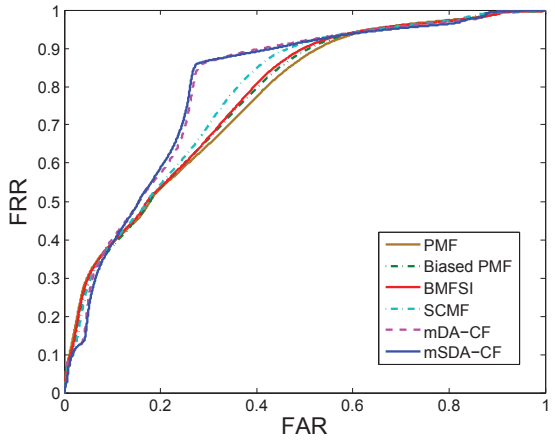

Figure 2: ROC curves of our approach and compared methods on Advertising dataset (d=10).

The advertising dataset used in our experiments is very sparse, which contains 880,569 click responses (density: 0.27%). We use the first 50% click responses for training, and the remaining data for testing. Following [44], we use the receiver operating characteristic (ROC) curve and the area under ROC (AUC-ROC) as our evaluation metrics. The true positive rate (TPR) and false positive rate (FPR) used for generating ROC curves are defined as follows:

$$TPR = \frac{TP}{TP+FN}, \quad FPR = \frac{FP}{FP+TN}, \tag{22}$$

where TP represents the true positives, FN represents the false negatives, TN represents the true negatives, and FP represents the false positives.

We evaluate the performance of each compared method on the Advertising dataset. The major parameters α, β and λ are set to 0.8, 0.02 and 0.12, respectively. Two settings are utilized, by setting the latent factor dimension to 10 and 20, respectively. Fig. 2 shows the ROC curves of PMF, Biased PMF, BMFSI, SCMF, mDA-CF and mSDA-CF. Table 7 shows the corresponding AUC of all compared methods. We can observe that our approaches obtain higher AUC than other methods, which demonstrates the effectiveness of our framework.

5.4 Discussion

So far, we have seen that our approach outperforms the existing approaches on the different datasets. We also analyze the convergence property and parameter settings of our approach on the

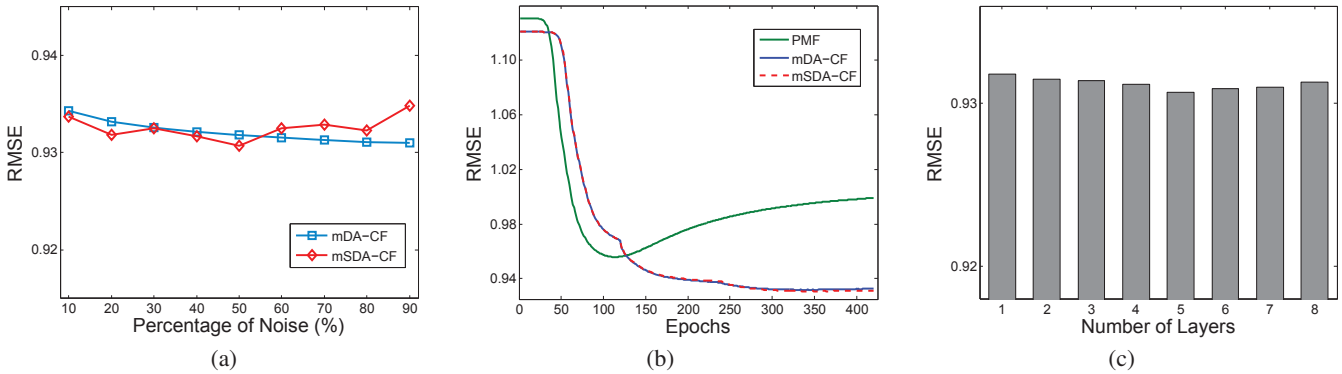

Figure 3: Experimental analysis on MovieLens-100K dataset: (a) RMSE with different level of noise; (b) RMSE with epochs; (c) RMSE of mSDA-CF with different number of layers.

Table 7: AUC of compared methods on Advertising dataset.

Method	AUC ($d = 10$)	AUC ($d = 20$)
PMF [4]	0.7651	0.7716
Biased PMF [41]	0.7692	0.7724
BMFSI [26]	0.7720	0.7805
SCMF [23]	0.7782	0.7866
mDA-CF (Ours)	0.7961	0.8023
mSDA-CF (Ours)	**0.8057**	**0.8115**

MovieLens-100K dataset. In Fig. 3(a), we show the RMSE of mDA-CF and mSDA-CF with different levels of noise. We observe an interesting phenomena that the RMSE of mDA-CF slightly decreases when increasing the noise level for input samples at first; mSDA-CF achieves the best performance when adding 50% noise. However, when the percentage of noise is larger than 50%, mDA-CF outperforms mSDA-CF. It shows that the latent factors learned from multi-layer models might be unreliable if there are too much noise contained in the input samples.

Fig. 3(b) shows the RMSE in different iterations of PMF and our approaches. We see that PMF overfits the data after 100 epochs. Although our approaches have larger RMSE from epoch 40 to epoch 120, they keep reducing the RMSE even after 400 epochs. It shows that our approach enjoys better stability. Also, by incorporating the side information of users and items, the learned model has a good generalization ability.

Another important property in our approach is the setting of stacked layers. Fig. 3(c) shows the RMSE of mSDA-CF with different number of layers. In general, stacking multiple auto-encoder will lead to better performance. But Fig. 3(c) shows that the improvement becomes marginal when there are more than 5 layers. The reasons are two-fold. First, the model capacity is related to the size of training data. In the experiments, training a very complicated deep model (with many layers) with the employed datasets may be unreasonable. Secondly, accessing rich features of users is always a challenging issue in reality. In our model, these features are the inputs of deep models. Our model achieves better performance than existing works by using these features. However, the features are not extensive enough to train deep models. In a word, this parameter should be carefully tuned in practice.

6. CONCLUSIONS

In this paper, we propose a deep collaborative filtering (DCF) framework, which bridges matrix factorization and deep feature learning. DCF is a hybrid collaborative filtering model, as it learns effective latent factors from both user-item ratings and side information. Using this framework, we present the mDA-CF and mSDA-CF approaches by incorporating the probabilistic matrix factorization and marginalized denoising auto-encoders. We also design efficient optimization algorithms to solve the models. Extensive experimental results on the MovieLens-100K, MovieLens-1M, Book-Crossing and Advertising datasets demonstrate the effectiveness of the latent factors learned by our model. Our mDA-CF and mSDA-CF approaches outperform related methods on the tasks of movie recommendation, book recommendation and response prediction. They also achieve better performance than the existing deep learning based collaborative filtering method such as UI-RBM. In addition, the convergence property and parameter settings of our model are discussed in the experiments.

A part of the future work is to extend other deep learning and matrix factorization methods using our DCF framework and evaluate their performance for collaborative filtering. Another future direction is to apply the distributed optimization algorithms to further reduce the computational costs of our algorithms.

Acknowledgements

This research is supported in part by the National Science Foundation (NSF) CNS award 1314484, Office of Naval Research (ONR) award N00014-12-1-1028, ONR Young Investigator Award N00014-14-1-0484, Naval Postgraduate School (NPS) award N00244-15-1-0041 and U.S. Army Research Office Young Investigator Award W911NF-14-1-0218.

7. REFERENCES

[1] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. Artificial Intellegence*, 2009.

[2] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[3] Sotirios Chatzis. Nonparametric bayesian multitask collaborative filtering. In *CIKM*, pages 2149–2158, 2013.

[4] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[5] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. In *WSDM*, pages 287–296, 2011.

[6] Ajit Paul Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *KDD*, pages 650–658, 2008.

[7] Ajit Paul Singh and Geoffrey J. Gordon. A bayesian matrix factorization model for relational data. *CoRR*, abs/1203.3517, 2012.

[8] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[9] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[10] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, pages 791–798, 2007.

[11] Kostadin Georgiev and Preslav Nakov. A non-iid framework for collaborative filtering with restricted boltzmann machines. In *ICML*, pages 1148–1156, 2013.

[12] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *ACM MM*, pages 627–636, 2014.

[13] Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *NIPS*, pages 2643–2651, 2013.

[14] Tran The Truyen, Dinh Q. Phung, and Svetha Venkatesh. Ordinal boltzmann machines for collaborative filtering. In *UAI*, pages 548–556, 2009.

[15] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. *CoRR*, abs/1409.2944, 2014.

[16] Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *ICML*, 2012.

[17] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B Kantor. *Recommender systems handbook*, volume 1. Springer, 2011.

[18] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, page 35, 2007.

[19] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322, 2010.

[20] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2001.

[21] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887, 2008.

[22] Minjie Xu, Jun Zhu, and Bo Zhang. Fast max-margin matrix factorization with data augmentation. *ICML*, 2013.

[23] Jianping Shi, Naiyan Wang, Yang Xia, Dit-Yan Yeung, Irwin King, and Jiaya Jia. SCMF: sparse covariance matrix factorization for collaborative filtering. In *IJCAI*, 2013.

[24] Ryan Prescott Adams, George E. Dahl, and Iain Murray. Incorporating side information in probabilistic matrix factorization with gaussian processes. In *UAI*, pages 1–9, 2010.

[25] Tong Zhao, Julian J. McAuley, and Irwin King. Leveraging social connections to improve personalized ranking for collaborative filtering. In *CIKM*, pages 261–270, 2014.

[26] Ian Porteous, Arthur U. Asuncion, and Max Welling. Bayesian matrix factorization with side information and dirichlet process mixtures. In *AAAI*, 2010.

[27] Yong-Deok Kim and Seungjin Choi. Scalable variational bayesian matrix factorization with side information. In *AISTATS*, pages 493–502, 2014.

[28] Sunho Park, Yong-Deok Kim, and Seungjin Choi. Hierarchical bayesian matrix factorization with side information. In *IJCAI*, 2013.

[29] Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Can Zhu. Personalized recommendation via cross-domain triadic factorization. In *WWW*, pages 595–606, 2013.

[30] Aditya Krishna Menon, Krishna Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *KDD*, pages 141–149, 2011.

[31] Sheng Li, Jaya Kawale, and Yun Fu. Predicting user behavior in display advertising via dynamic collective matrix factorization. In *SIGIR*, 2015.

[32] Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Wei Cao. Deep modeling of group preferences for group-based recommendation. In *AAAI*, pages 1861–1867, 2014.

[33] Yuanxin Ouyang, Wenqi Liu, Wenge Rong, and Zhang Xiong. Autoencoder-based collaborative filtering. In *ICONIP*, pages 284–291, 2014.

[34] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887. ACM, 2008.

[35] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[36] Koray Kavukcuoglu, MarcâĂŹAurelio Ranzato, Rob Fergus, and Yann Le-Cun. Learning invariant features through topographic filter maps. In *CVPR*, pages 1605–1612. IEEE, 2009.

[37] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS*, pages 1096–1104, 2009.

[38] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103. ACM, 2008.

[39] Minmin Chen, Kilian Q. Weinberger, Fei Sha, and Yoshua Bengio. Marginalized denoising auto-encoders for nonlinear representations. In *ICML*, pages 1476–1484, 2014.

[40] Zhi-Qin Yu, Xing-Jian Shi, Ling Yan, and Wu-Jun Li. Distributed stochastic ADMM for matrix factorization. In *CIKM*, pages 1259–1268, 2014.

[41] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.

[42] Hao Ma. An experimental study on implicit social recommendation. In *SIGIR*, pages 73–82, 2013.

[43] Fangfang Li, Guandong Xu, and Longbing Cao. Coupled item-based matrix factorization. In *WISE*, pages 1–14, 2014.

[44] Amr Ahmed, Abhimanyu Das, and Alexander J. Smola. Scalable hierarchical multitask learning algorithms for conversion optimization in display advertising. In *WSDM*, pages 153–162, 2014.