Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems

Xin Dong, Lei Yu, Zhonghuo Wu, Yuxia Sun, Lingfeng Yuan, Fangxi Zhang
Ctrip Travel Network Technology (Shanghai) Co., Limited.
Shanghai, P.R.China
{dongxin, yu_lei, zh_wu, yx_sun, lfyuan, fxzhang}@ctrip.com

Abstract

Collaborative filtering(CF) is a widely used approach in recommender systems to solve many real-world problems. Traditional CF-based methods employ the user-item matrix which encodes the individual preferences of users for items for learning to make recommendation. In real applications, the rating matrix is usually very sparse, causing CF-based methods to degrade significantly in recommendation performance. In this case, some improved CF methods utilize the increasing amount of side information to address the data sparsity problem as well as the cold start problem. However, the learned latent factors may not be effective due to the sparse nature of the user-item matrix and the side information. To address this problem, we utilize advances of learning effective representations in deep learning, and propose a hybrid model which jointly performs deep users and items' latent factors learning from side information and collaborative filtering from the rating matrix. Extensive experimental results on three real-world datasets show that our hybrid model outperforms other methods in effectively utilizing side information and achieves performance improvement.

Introduction

In recent years, with the growing number of choices available online, recommender systems are becoming more and more indispensable. The goal of recommender systems is to help users in identifying the items that best fit their personal tastes from a large repository of items. Besides, many commerce companies have been using recommender systems to target their customers by recommending items. Over the years, various algorithms for recommender systems have been developed. Such algorithms can roughly be categorized into two groups (Shi, Larson, and Hanjalic 2014): content-based and collaborative filtering(CF) based methods. Content-based methods (Lang 1995) utilize user profile or item content information for recommendation. CF-based methods (Salakhutdinov and Mnih 2011), on the other hand, ignore user or item content information and use the past activities or preferences, such as user buying/viewing history or user ratings on items, to recommendation. Nevertheless, CF-based methods are often preferred to content-based methods because of their impressive performance (Su and Khoshgoftaar 2009).

The most successful approach among CF-based methods is to learn effective latent factors directly by matrix factorization technique from the user-item rating matrix (Koren et al. 2009). However, the rating matrix is often very sparse in real world, causing CF-based methods to degrade significantly in learning the appropriate latent factors. In particular, this phenomenon occurs seriously in online travel agent(OTA) websites such as Ctrip.com, since user access these websites with lower frequency. Moreover, another limitation for CF-based methods is how to provide recommendations when a new item arrives in the system, which is also known as the cold start problem. The reason of the existence about cold start is that the systems cannot recommend new items which have not yet receive rating information from users.

In order to overcome the cold start and data sparsity problems, it is inevitable for CF-based methods to exploit additional sources of information about the users or items, also known as the side information, and hence hybrid CF methods have gained popularity in recent years (Shi, Larson, and Hanjalic 2014). The side information can be obtained from user profile and item content information, such as demographics of users, properties of items, etc. Some hybrid CF-based methods (Singh and Gordon 2008; Nickel, Tresp, and Kriegel 2011; Wang and Blei 2011) have integrated side information into matrix factorization to learn effective latent factors. However, these methods employ the side information as regularizations and the learned latent factors are often not effective especially when the rating matrix and side information are very spare (Agarwal, Chen, and Long 2011). Therefore, it is highly desirable to realize this latent factor learning problem from such datasets.

Recently, one of the powerful methods to learn effective representations is deep learning (Hinton and Salakhutdinov 2006; Hinton, Osindero, and Teh 2006). Thus, with large-scale ratings and rich additional side information, it is nature to integrate deep learning in recommender systems to learn latent factors. Thereby, some researches have made use of deep learning directly for the task of collaborative filtering. Work (Salakhutdinov, Mnih, and Hinton 2007) employs restricted Boltzmann machines to perform CF. Although this method combines deep learning and CF, it does not incorporate side information, which is crucial for accurate recommendation. Moreover, work (Van den Oord, Diele-

man, and Schrauwen 2013; Wang and Wang 2014) directly uses convolutional neural network(CNN) or deep belief network(DBN) to obtain latent factors for content information, but they are content-based methods which only infer latent factors for items and the methods are especially fit for music datasets. Furthermore, work (Wang, Wang, and Yeung 2015; Li, Kawale, and Fu 2015) utilizes Bayesian stacked denoising auto-encoders(SDAE) or marginalized SDAE to CF but requires learning of a large number of manually adjusted hyper parameters.

In this paper, to address the challenges above, we propose a hybrid collaborative filtering model with deep structure for recommender systems. We first present a novel deep learning model called additional stacked denoising autoencoder(aSDEA), which extends the stacked denoising autoencoder to integrate additional side information into the inputs, and then overcomes cold start problem and data sparsity problem. With this, we then present our hybrid model which tightly couples deep representation learning for the additional side information and collaborative filtering for the ratings matrix. Experiments show that our hybrid model significantly outperforms the state of the art. Specifically, the main contributions of this paper can be summarized as the following three aspects:

- We propose a hybrid collaborative filtering model, which integrates deep presentation learning and matrix factorization. It simultaneously extracts effective latent factors from side information and captures the implicit relationship between users and items.
- We present a novel deep learning model aSDAE, which is a variant of SDAE and can integrate the side information into the learned latent factors efficiently.
- We conduct experiments on three real-world datasets to evaluate the effectiveness of our hybrid model. Experimental results show that our hybrid model outperforms four state-of-art methods in terms of root mean squared error(RMSE) and recall metrics.

Preliminaries

In this section, we start with formulating the problem discussed in this paper, and then have a brief view on matrix factorization.

Problem Definition

Similar to some existing works(Hu, Koren, and Volinsky 2008), this paper also takes implicit feedback as training and testing data to complete the recommendation task. In a standard recommendation setting, we have m users, n items, and an extremely sparse rating matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$. Each entry $\mathbf{R}_{ij}$ of $\mathbf{R}$ corresponds to user i's rating on item j. If $\mathbf{R}_{ij} \neq 0$, it means the rating about user i on item j is observed, otherwise unobserved. Each user i can be represented by a partially observed vector $\mathbf{s}_i^{(u)} = (\mathbf{R}_{i1}, ..., \mathbf{R}_{in}) \in \mathbb{R}^n$. Identically, each item j can be represented by a partially observed vector $\mathbf{s}_j^{(i)} = (\mathbf{R}_{1j}, ..., \mathbf{R}_{mj}) \in \mathbb{R}^m$. Moreover, the additional side information matrix of user and item are denoted by $\mathbf{X} \in \mathbb{R}^{m \times p}$ and $\mathbf{Y} \in \mathbb{R}^{n \times q}$, respectively.

Let $u_i, v_j \in \mathbb{R}^k$ be user i's latent factor vector and item j's latent factor vector respectively, where k is the dimensionality of the latent space. Therefore, the corresponding matrix forms of latent factors for users and items are $\mathbf{U} = u_{1:m}$ and $\mathbf{V} = v_{1:n}$, respectively. Given the sparse rating matrix $\mathbf{R}$ and the side information matrix $\mathbf{X}$ and $\mathbf{Y}$, the goal is to learn user latent factors $\mathbf{U}$ and item latent factors $\mathbf{V}$, and hence to predict the missing ratings in $\mathbf{R}$.

Matrix Factorization

An effective collaborative filtering approach is matrix factorization (Koren et al. 2009). By factorizing the user-item interactions matrix, matrix factorization can map both users and items to a joint latent factor space. Therefore, user-item interactions are modeled as inner products in that space. Formally, matrix factorization decomposes the original rating matrix $\mathbf{R}$ into two low-rank matrices $\mathbf{U}$ and $\mathbf{V}$ consisting of the user and item latent factor vectors respectively, such that $\mathbf{R} \approx \mathbf{UV}$. Given the latent factor vectors for users and items, a user's rating for a movie is predicted by the inner product of those vectors.

The objective function of matrix factorization can be written as:

$$\arg\min_{\mathbf{U},\mathbf{V}} \mathcal{L}(\mathbf{R}, \mathbf{U}\mathbf{V}^T) + \lambda(||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2),$$

where $\mathcal{L}(\cdot, \cdot)$ is the loss function that measures the distance between two matrices with the same size, the last two terms are the regularizations used to avoid overfitting and $|| \cdot ||_F$ denotes the Frobenius norm. By specifying different $\mathcal{L}(\cdot, \cdot)$, many matrix factorization models have been proposed, for example, non-negative matrix factorization (Lee and Seung 2001), probabilistic matrix factorization (Salakhutdinov and Mnih 2011), Bayesian probabilistic matrix factorization (Salakhutdinov and Mnih 2008), max-margin matrix factor (Srebro, Rennie, and Jaakkola 2004), etc.

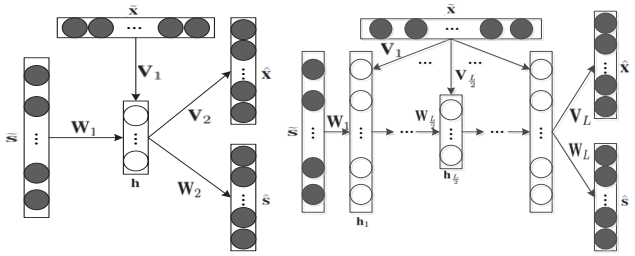
When side information are available, some matrix factorization models generate a rating from the product of latent factor vectors which contain additional information about users or items. Various models show that additional side information can act as a useful informative prior that can significantly improve results (Porteous, Asuncion, and Welling 2010; Singh and Gordon 2008).

Additional Stacked Denoising Autoencoder

In this section we first provide a introduction of additional denoising autoencoder and then give a detailed description of additional stacked denoising autoencoder(aSDAE).

Additional Denoising Autoencoder

An autoencoder is a specific form of neural network, which consists of an encoder and a decoder component. The encoder $g(\cdot)$ takes a given input $\mathbf{s}$ and maps it to a hidden representation $g(\mathbf{s})$, while the decoder $f(\cdot)$ maps this hidden representation back to a reconstructed version of $\mathbf{s}$, such that $f(g(\mathbf{s})) \approx \mathbf{s}$. The parameters of the autoencoder are learned to minimize the reconstruction error, measured by some loss



(a) Additional Denoising Autoencoder(aDAE) (b) Additional Stacked Denoising Autoencoder(aSDAE)

Figure 1: The models of aDAE and aSDAE

$\mathcal{L}(\mathbf{s}, f(g(\mathbf{s})))$. However, denoising autoencoders(DAE) incorporate a slight modification to this setup, which reconstructs the input from a corrupted version with the motivation of learning a more effective representation from the input(Vincent et al. 2008). A denoising autoencoder is trained to reconstruct the original input $\mathbf{s}$ from its corrupted version $\tilde{\mathbf{s}}$ by minimizing $\mathcal{L}(\mathbf{s}, f(g(\tilde{\mathbf{s}})))$. Usually, choices of corruption include additive isotropic Gaussian noise or binary masking noise (Vincent et al. 2008). Moreover, various types of autoencoders have been developed in several domains to show promising results(Chen et al. 2012; Lee et al. 2009).

In this paper, we extend the denoising autoencoder to integrate additional side information into the inputs, as shown in Figure 1(a). Given a sample set $\mathbf{S} = [\mathbf{s}_1, ..., \mathbf{s}_n]$, the corresponding side information set $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]$, additional denoising autoencoder(aDAE) considers a random corruptions over $\mathbf{S}$ and $\mathbf{X}$ to obtain $\tilde{\mathbf{S}}$ and $\tilde{\mathbf{X}}$. It then encoders and decoders the inputs as follows:

$$\begin{aligned}\mathbf{h} &= g(\mathbf{W}_1\tilde{\mathbf{s}} + \mathbf{V}_1\tilde{\mathbf{x}} + \mathbf{b}_h)\\ \hat{\mathbf{s}} &= f(\mathbf{W}_2\mathbf{h} + \mathbf{b}_{\hat{s}})\\ \hat{\mathbf{x}} &= f(\mathbf{V}_2\mathbf{h} + \mathbf{b}_{\hat{x}}),\end{aligned}$$

where $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{x}}$ represent the corrupted version of the original inputs $\mathbf{s}$ and $\mathbf{x}$, $\hat{\mathbf{s}}$ and $\hat{\mathbf{x}}$ represent the reconstructions of $\mathbf{s}$ and $\mathbf{x}$, $\mathbf{h}$ represents the hidden latent representation of the inputs, $\mathbf{W}$ and $\mathbf{V}$ are weight matrix, $\mathbf{b}$ is bias vector, and $g(\cdot)$ and $f(\cdot)$ are activation functions such as $tanh(\cdot)$.

The objective function considers the losses between all the inputs and their reconstructions. Then, an aDAE solves the following optimization problem:

$$\arg \min_{\{\mathbf{W}_l\},\{\mathbf{V}_l\},\{\mathbf{b}_l\}} \alpha||\mathbf{S} - \hat{\mathbf{S}}||_F^2 + (1-\alpha)||\mathbf{X} - \hat{\mathbf{X}}||_F^2 + \lambda(\sum_l ||\mathbf{W}_l||_F^2 + ||\mathbf{V}_l||_F^2), \quad (1)$$

where α is a trade-off parameter which balances the outputs, and λ is a regularization parameter.

Additional Stacked Denoising Autoencoder

Existing literatures have shown that multiple layers stacked together can generate rich representations in hidden layers, and therefore leads to better performance for various tasks (Rifai et al. 2011; Chen et al. 2012; Kavukcuoglu et al. 2009; Glorot, Bordes, and Bengio 2011). The stacked denoising autoencoder(SDAE) stacks several DAEs together to create higher-level representations (Vincent et al. 2010). Inspired by the stacked denoising autoencoder, we stack multiple aDAE together to form an additional stacked denoising autoencoder(aSDAE). The model of aSDAE is shown in Figure 1(b) and the generative process is presented as follows:

- For each hidden layer $l \in \{1, ..., L-1\}$ of the aSDAE model, the hidden representation $\mathbf{h}_l$ is computed as:

$$\mathbf{h}_l = g(\mathbf{W}_l\mathbf{h}_{l-1} + \mathbf{V}_l\tilde{\mathbf{x}} + \mathbf{b}_l),$$

  where $\mathbf{h}_0 = \tilde{\mathbf{s}}$ is one of the corrupted inputs.

- For the output layer L, the outputs are computed as:

$$\begin{aligned}\hat{\mathbf{s}} &= f(\mathbf{W}_L\mathbf{h}_L + \mathbf{b}_{\hat{s}})\\ \hat{\mathbf{x}} &= f(\mathbf{V}_L\mathbf{h}_L + \mathbf{b}_{\hat{x}})\end{aligned}$$

Note that the first $L/2$ layers of the model act as an encoder and the last $L/2$ layers act as a decoder. The aSDAE employs a deep network to reconstruct the inputs and minimizes the squared loss between inputs and their reconstructions. The objective function for aSDAE is similar with Equation (1). Accordingly, we can learn $\mathbf{W}_l$, $\mathbf{V}_l$ and $\mathbf{b}_l$ for each layer using the back-propagation algorithm. In our aSDAE model, we assume that only one hidden layer should be close to the latent factor and the latent factor vector is generated from the $L/2$ layer, given the total number of layers is L.

A Hybrid Collaborative Filtering Model

In some CF-based methods, the main challenges are to infer effective and high-level latent factor vectors for users and items from raw inputs. MF-based methods are able to meet the requirements so as to capture the implicit relationship between the users and items. However, they suffer from the cold start and data sparsity problems. Moreover, deep learning models have been shown to be highly effective in discovering high-level hidden representations from the raw input data for a variety of tasks (Shen et al. 2014; Li, Kawale, and Fu 2015; Wang, Wang, and Yeung 2015). Therefore, it is straightforward to take over the expressive ability from deep learning to improve the collaborative filtering algorithms.

In this section, we propose a hybrid collaborative filtering model which unifies our aSDAE model with matrix factorization for recommender systems.

Overview

The proposed model is a hybrid model, which makes use of both rating matrix and side information and combines aSDAE and matrix factorization together. Matrix factorization is a widely used model-based CF method with excellent scalability and accuracy, and aSDAE is a powerful way to extract high-level representations from raw inputs. The combination of this two models leverages their benefits for learning more expressive models.

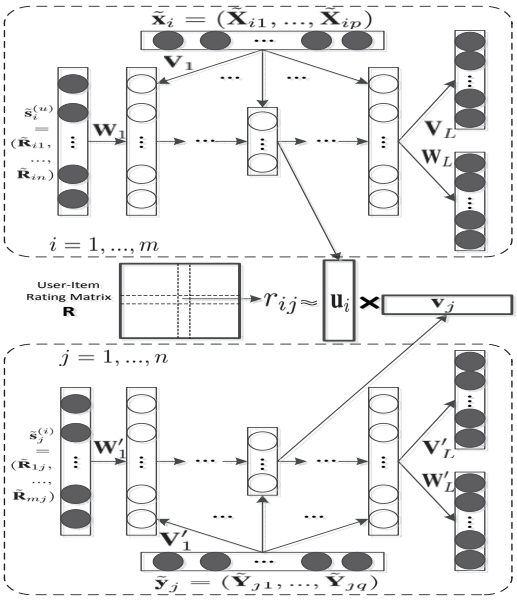


Figure 2: The structure of the proposed hybrid model. The model contains three components: the upper component and the lower component are two aSDAEs which extract latent factor vectors for users and items respectively; the middle component decomposes the rating matrix $\mathbf{R}$ into two latent factor matrices.

Given the user-item rating matrix $\mathbf{R}$, we first transform $\mathbf{R}$ into the set $\mathbf{S}^{(u)}$ containing m instances $\{\mathbf{s}_1^{(u)}, ..., \mathbf{s}_m^{(u)}\}$, where $\mathbf{s}_i^{(u)} = \{\mathbf{R}_{i1}, ..., \mathbf{R}_{in}\}$ is the n-dimensional feedback vector of user i on all the items. Similarly, we can obtain set $\mathbf{S}^{(i)}$ with n instances $\{\mathbf{s}_1^{(i)}, ..., \mathbf{s}_n^{(i)}\}$, where $\mathbf{s}_j^{(i)} = \{\mathbf{R}_{1j}, ..., \mathbf{R}_{mj}\}$ is the m-dimensional feedback vector of item j rated by all the users. Our hybrid model learns user and item latent factors(i.e., $\mathbf{U}$ and $\mathbf{V}$) from $\mathbf{R}$, $\mathbf{S}^{(u)}$, $\mathbf{S}^{(i)}$ and the additional side information(i.e., $\mathbf{X}$ and $\mathbf{Y}$) through the following optimization objective:

$$\arg\min_{\mathbf{U},\mathbf{V}} \mathcal{L}_R(\mathbf{R}, \mathbf{U}\mathbf{V}^T) + \lambda(||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2) + \beta\mathscr{L}(\mathbf{S}^{(u)}, \mathbf{X}, \mathbf{U}) + \delta\mathscr{L}(\mathbf{S}^{(i)}, \mathbf{Y}, \mathbf{V}), \quad (2)$$

where $\mathcal{L}_R(\cdot,\cdot)$ is the loss function for decomposing the rating matrix $\mathbf{R}$ into two latent factor matrices $\mathbf{U}$ and $\mathbf{V}$, $\mathscr{L}(\cdot,\cdot,\cdot)$ is the function that connects the user or item side information with the latent factors, β and δ are the trade-off parameters and λ is a regularization parameter. Note that, the last two terms in Equation (2) devised using our aSDAE model which extracts latent factor matrix from the rating matrix and additional side information.

Our Hybrid Model

Let $\mathbf{S}^{(u)} \in \mathbb{R}^{m\times n}$ and $\mathbf{S}^{(i)} \in \mathbb{R}^{n\times m}$ denote the matrices obtained in the above section, and let $\tilde{\mathbf{S}}^{(u)}$ and $\tilde{\mathbf{S}}^{(i)}$ denote their corrupted versions respectively. Moreover, $\mathbf{X} \in \mathbb{R}^{m\times p}$ and $\mathbf{Y} \in \mathbb{R}^{n\times q}$ are the additional side information matrices about users and items respectively, and the corresponding corrupted versions are $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$. Obviously, Figure 2 illustrates our hybrid collective filtering model. It indicates that the inputs of the hybrid model are $\tilde{\mathbf{S}}^{(u)}$, $\tilde{\mathbf{S}}^{(i)}$, $\tilde{\mathbf{X}}$, $\tilde{\mathbf{Y}}$ and $\mathbf{R}$. As shown in Equation (2), the first term is the loss function of matrix factorization to decompose the rating matrix $\mathbf{R}$ into user and item latent factor matrices, i.e.,

$$\mathcal{L}_R(\mathbf{R}, \mathbf{U}\mathbf{V}^T) = \sum_{i,j} \mathbf{I}_{ij}(\mathbf{R}_{ij} - \mathbf{u}_i\mathbf{v}_j^T)^2$$

where $\mathbf{I}$ is an indicator matrix indicating the non-empty entities in $\mathbf{R}$. The last two terms are the loss functions of our aSDAE models which extract latent factors from the hidden layers for users and items respectively. For simplicity, we set β and δ to 1 in Equation (2). Therefore, the objective function of our hybrid model is formulated as follows:

$$\begin{aligned} L = \; & \sum_{i,j} \mathbf{I}_{ij}(\mathbf{R}_{ij} - \mathbf{u}_i\mathbf{v}_j^T)^2 + \alpha_1 \sum_i (\mathbf{s}_i^{(u)} - \hat{\mathbf{s}}_i^{(u)})^2 \\ & + (1-\alpha_1)\sum_i (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2 + \alpha_2 \sum_j (\mathbf{s}_j^{(i)} - \hat{\mathbf{s}}_j^{(i)})^2 \\ & + (1-\alpha_2)\sum_j (\mathbf{y}_j - \hat{\mathbf{y}}_j)^2 + \lambda \cdot f_{reg} \end{aligned} \quad (3)$$

where $\alpha_1, \alpha_2$are trade-off parameters, and f_{reg} are the regularization terms that prevent overfitting, i.e.,

$$\begin{aligned} f_{reg} = & \sum_i ||\mathbf{u}_i||_F^2 + \sum_j ||\mathbf{v}_j||_F^2 + \sum_l (||\mathbf{W}_l||_F^2 + ||\mathbf{V}_l||_F^2 \\ & + ||\mathbf{b}_l|| + ||\mathbf{W}'_l||_F^2 + ||\mathbf{V}'_l||_F^2 + ||\mathbf{b}'_l||_F^2), \end{aligned}$$

$\mathbf{W}_l, \mathbf{V}_l$ and $\mathbf{W}'_l, \mathbf{V}'_l$ are the weight matrices for two aSDAEs at layer l, $\mathbf{b}_l$ and $\mathbf{b}'_l$ are the corresponding bias vectors.

Generally, the middle layers of two aSDAEs server as bridges between the ratings and additional side information. These two middle layers are the key that enables our hybrid model to simultaneously learn effective latent factors and capture the the similarity and relationship between users and items.

Optimization

Although the optimization of the objective function is not jointly convex in all the variables, it is convex to each of them when fixing the others. Therefore, we can alternately optimize for each of the variables in the above objective function.

For $\mathbf{u}_i$ and $\mathbf{v}_j$, we use stochastic gradient descent(SGD) algorithm to learn these latent factors. For simplicity, we let $L(\mathbf{U}, \mathbf{V})$ denote the objective function when other variables irrelevant to $\mathbf{U}$ and $\mathbf{V}$ are fixed. Therefore, the update rules are:

$$\mathbf{u}_i = \mathbf{u}_i - \eta \frac{\partial}{\partial \mathbf{u}_i} L(\mathbf{U}, \mathbf{V}),$$
$$\mathbf{v}_j = \mathbf{v}_j - \eta \frac{\partial}{\partial \mathbf{v}_j} L(\mathbf{U}, \mathbf{V}),$$

where η is the learning rate, and the detail gradients are as follows:

$$\frac{\partial}{\partial \mathbf{u}_i} L(\mathbf{U}, \mathbf{V}) = \alpha \sum_i (\mathbf{s}_i^{(u)} - \hat{\mathbf{s}}_i^{(u)}) \frac{\partial \hat{\mathbf{s}}_i^{(u)}}{\partial \mathbf{u}_i} + \lambda \mathbf{u}_i$$
$$+ (1-\alpha) \sum_i (\mathbf{x}_i - \hat{\mathbf{x}}_i) \frac{\partial \hat{\mathbf{x}}_i}{\partial \mathbf{u}_i} - \sum_{i,j \in \mathbf{I}} (\mathbf{R}_{ij} - \mathbf{u}_i \mathbf{v}_j^T) \mathbf{v}_j$$
$$\frac{\partial}{\partial \mathbf{v}_j} L(\mathbf{U}, \mathbf{V}) = \alpha \sum_j (\mathbf{s}_j^{(i)} - \hat{\mathbf{s}}_j^{(i)}) \frac{\partial \hat{\mathbf{s}}_j^{(i)}}{\partial \mathbf{v}_j} + \lambda \mathbf{v}_j$$
$$+ (1-\alpha) \sum_j (\mathbf{y}_j - \hat{\mathbf{y}}_j) \frac{\partial \hat{\mathbf{y}}_j}{\partial \mathbf{v}_j} - \sum_{i,j \in \mathbf{I}} (\mathbf{R}_{ij} - \mathbf{u}_i \mathbf{v}_j^T) \mathbf{u}_i.$$

Note that, we set α_1 equal to α_2 in Equation (3) for simplicity. Moreover, given $\mathbf{U}$ and $\mathbf{V}$, we can learn the weight matrices and biases for each layer using the popular back-propagation learning algorithm. By alternating the update of variables, a local optimum for L can be found. Nevertheless, we can use some common techniques such as using a momentum term to alleviate the local optimum problem.

Prediction

After the latent factors for each user and item are learned, we approximate the predicted rating $\hat{\mathbf{R}}_{ij}$ as: $\hat{\mathbf{R}}_{ij} \approx \mathbf{u}_i \mathbf{v}_j^T$, and then a list of ranked items is generated for each user based on these prediction ratings.

Experiments

In this section, we evaluate the performance of our hybrid model with three real-world datasets from different domains, and compare our hybrid model with four state-of-art algorithms.

Datasets

We use three datasets from different real-world domains, two from MovieLens and one from Book-Crossing dataset, for our experiments. The first two datasets, MovieLens-100K and MovieLens-1M, are commonly used for evaluating the performance of recommender systems (Wang, Shi, and Yeung 2015; Li, Kawale, and Fu 2015). The MovieLens-100K dataset contains 100K ratings from 943 users on 1682 movies, and the MovieLens-1M dataset contains more than 1 million ratings from 6040 users on 3706 movies. Each rating is an integer between 1 and 5. We binarize explicit data by keeping the ratings of four or higher and interpret them as implicit feedback. Therefore, MovieLens-1M is much sparser as only 2.57% of its user-item matrix entries contain ratings but MovieLens-100K has ratings in 3.49% of its user-item matrix entries. Moreover, we extract the user and item information provided by the datasets to construct the additional matrices $\mathbf{X}$ and $\mathbf{Y}$ respectively. To summarize, the user side information contains the user's ID, age, gender, occupation and zipcode are encoded into a binary valued vector of length 1943. Identically, the item side information contains the item's title, release data and 18 categories of movie genre are encoded into a binary valued vector of length 1822.

The last dataset, Book-Crossing dataset, contains 1149780 ratings from 278858 users on 271379 books. The rating is expressed on a scale from 0 to 10 with the higher values denoting higher appreciation. However, we binarize explicit data by keeping the ratings of six or higher and interpret them as implicit feedback. This leads to the user-item matrix with a sparsity of 99.99%. Some attributes for users and books are also provided in this dataset. The user and item additional matrices are generated as the above datasets, and the lengths of the two binary vectors are 1973 and 3679.

Evaluation Metric

We employ the root mean squared error(RMSE) as one of the evaluation metrics,

$$\text{RMSE} = \sqrt{\frac{1}{|\mathbf{T}|} \sum_{\mathbf{R}_{ij} \in \mathbf{T}} (\mathbf{R}_{ij} - \hat{\mathbf{R}}_{ij})^2},$$

where $\mathbf{R}_{ij}$ is the rating of user i on item j, $\hat{\mathbf{R}}_{ij}$ denotes the corresponding predicted rating, $\mathbf{T}$ is the test set and $|\mathbf{T}|$ is the total number of ratings in the test set.

Similar to (Wang, Wang, and Yeung 2015; Wang and Blei 2011), we use recall as another evaluation metric since the rating information is in the form of implicit feedback (Hu, Koren, and Volinsky 2008; Rendle et al. 2009). Specifically, another common metric, precision, is not suited for implicit feedback. Because a zero rating in the user-item matrix may be due to the fact that the user is not interested in the item, or that the user is unaware of it. To evaluate our hybrid model, we sort the predicted ratings of all the items for each user, and then recommend the top K items to each user. The recall@K for each user is defined as follows:

$$\text{recall@}K = \frac{\text{number of items the user likes in top K}}{\text{total number of items the user likes}}.$$

The final metric result is the average recall over all users.

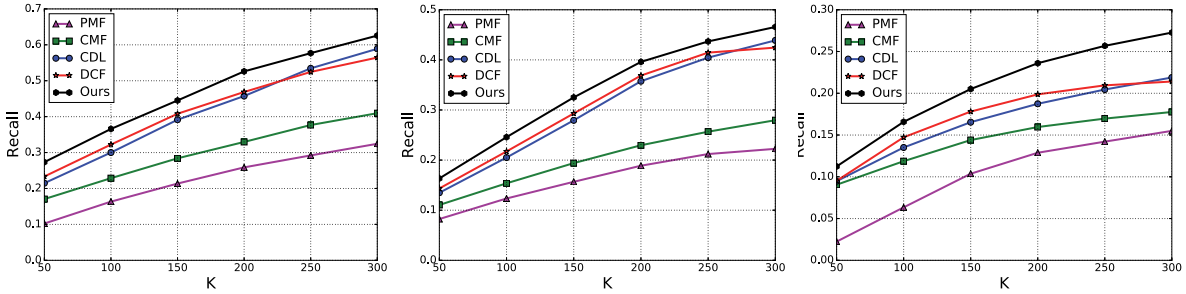
Baselines and Parameter Settings

In order to evaluate the performance of our model, we compare it with the following recommendation algorithms:

- **PMF.** Probabilistic Matrix Factorization (Salakhutdinov and Mnih 2011) is a model to factorize the user-item matrix to user and item factors. It assumes there exists Gaussion observation noise and Gaussian priors on the latent factor vectors.
- **CMF.** Collective Matrix Factorization (Singh and Gordon 2008) is a model which simultaneously factorizes multiple sources, including the user-item matrix and matrices containing the additional side information.
- **CDL.** Collaborative Deep Learning (Wang, Wang, and Yeung 2015) is a hierarchical deep Bayesian model to achieve deep representation learning for the item information and collaborative filtering for the user-item matrix.

Table 1: Average RMSE of compared models with different percentages of training data on three datasets

Model	MovieLen-100K			MovieLen-1M			Book-Crossing		
	60%	80%	95%	60%	80%	95%	60%	80%	95%
PMF	0.7024	0.5941	0.5673	0.6966	0.5715	0.5415	1.7534	1.2453	1.1896
CMF	0.6986	0.5881	0.5454	0.6758	0.5709	0.5382	1.5467	1.0563	0.9715
CDL	0.6601	0.5667	0.5213	0.6546	0.5435	0.5221	1.4465	0.9921	0.9652
DCF	0.6516	0.5516	0.5135	0.6635	0.5467	0.5335	1.3413	0.9784	0.9448
Ours	0.6436	0.5435	0.5079	0.6449	0.5236	0.5023	1.3206	0.9579	0.9244



(a) Recall on MovieLen-100K (b) Recall on MovieLen-1M (c) Recall on Book-Crossing

Figure 3: Performance comparison of PMF, CMF, CDL, DCF and Ours based on recall@K for the three datasets.

- **DCF.** Deep Collaborative Filtering (Li, Kawale, and Fu 2015) is a model which combines PMF with marginalized denoising stacked autoencoders to achieve recommendation.
- **Ours.** Our approach is proposed as described above. It is a hybrid collaborative filtering model which unifies our aSDAE model with matrix factorization.

For all the compared models, we train each compared method with different percentages(60%, 80% and 95%) of ratings. We randomly select the training set from each dataset, and use the remaining data as the test set. We repeat the evaluation five times with different randomly selected training sets and the average performance is reported. For our hybrid model, we set the parameters α, β and λ to 0.2, 0.8, and 0.01, respectively. The learning rate η used in SGD algorithm is set to 0.004. Similar to (Wang, Wang, and Yeung 2015), we use a masking noise with a noise level of 0.3 to get the corrupted inputs from the raw inputs. In terms of deep network architecture, the number of layers is set to 4 in our experiments. Moreover, the dimensionality of learned latent factors for user and item is set to 64.

Summary of Experimental Results

Table 1 shows the average RMSE of PMF, CMF, CDL, DCF and our hybrid model with different percentages of training data on the three datasets. We can observe from Table 1 that CMF, CDL, DCF and our hybrid model achieves better performance than PMF. It demonstrates the effectiveness of incorporating additional side information. Moreover, CDL, DCF and our hybrid model outperform PMF and CMF. That is, deep structures can admire better feature quality of side information. Furthermore, from Table 1, we can see that our hybrid model obtains lower RMSE than CDL and DCF, which validates the strengths of the latent factor vectors learned by our aSDAE models. Therefore, the RMSE metric demonstrates the effectiveness of our hybrid model.

Figure 3 shows the recall results that compare PMF, CMF, CDL, DCF and Our hybrid model using the three datasets. We can see that PMF is the worse model because of the lack of additional side information. Moreover, CMF performs worse than CDL, DCF and our hybrid model. This may be related to the discussion as in (Agarwal, Chen, and Long 2011), that is, when the side information is spare, CMF may not work well. Figure 3 also shows that our hybrid model achieves much better performance than CDL and DCF, as it takes advantage of our aSDAE models. Consequently, by seamlessly combining our aSDAE models for additional side information and matrix factorization for the user-item rating matrix, our hybrid model can handle both the sparse user-item rating matrix and the spare side information much better, and learn a much more effective latent factor for each user and item, and hence provides more accurate recommendation.

Conclusion

In this paper, we present a hybrid collaborative filtering model which bridges our aSDAE and matrix factorization. Our hybrid model can learn effective latent factors from both user-item rating matrix and side information for users and items. Moreover, the proposed deep learning model, aSDAE, is a variant of SDAE and can integrate the side information into the learned latent factors efficiently. Our experimental results present that our hybrid model outperforms other four state-of-art algorithms. As part of the future work, we will investigate other deep learning models to replace aSDAE

for boosting further performance, e.g., recurrent neural networks and convolutional neural networks.

References

Agarwal, D.; Chen, B.-C.; and Long, B. 2011. Localized factor models for multi-context recommendation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 609–617. ACM.

Chen, M.; Xu, Z.; Weinberger, K.; and Sha, F. 2012. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 513–520.

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

Hinton, G. E.; Osindero, S.; and Teh, Y.-W. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18(7):1527–1554.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, 263–272. IEEE.

Kavukcuoglu, K.; Fergus, R.; LeCun, Y.; et al. 2009. Learning invariant features through topographic filter maps. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 1605–1612. IEEE.

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

Lang, K. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning*, 331–339.

Lee, D. D., and Seung, H. S. 2001. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, 556–562.

Lee, H.; Pham, P.; Largman, Y.; and Ng, A. Y. 2009. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, 1096–1104.

Li, S.; Kawale, J.; and Fu, Y. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 811–820. ACM.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 809–816.

Porteous, I.; Asuncion, A. U.; and Welling, M. 2010. Bayesian matrix factorization with side information and dirichlet process mixtures. In *AAAI*.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, 452–461. AUAI Press.

Rifai, S.; Vincent, P.; Muller, X.; Glorot, X.; and Bengio, Y. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 833–840.

Salakhutdinov, R., and Mnih, A. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, 880–887. ACM.

Salakhutdinov, R., and Mnih, A. 2011. Probabilistic matrix factorization. In *NIPS*, volume 20, 1–8.

Salakhutdinov, R.; Mnih, A.; and Hinton, G. 2007. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, 791–798. ACM.

Shen, Y.; He, X.; Gao, J.; Deng, L.; and Mesnil, G. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 101–110. ACM.

Shi, Y.; Larson, M.; and Hanjalic, A. 2014. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)* 47(1):3.

Singh, A. P., and Gordon, G. J. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 650–658. ACM.

Srebro, N.; Rennie, J.; and Jaakkola, T. S. 2004. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, 1329–1336.

Su, X., and Khoshgoftaar, T. M. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009:4.

Van den Oord, A.; Dieleman, S.; and Schrauwen, B. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, 2643–2651.

Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, 1096–1103. ACM.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P.-A. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11(Dec):3371–3408.

Wang, C., and Blei, D. M. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 448–456. ACM.

Wang, X., and Wang, Y. 2014. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM international conference on Multimedia*, 627–636. ACM.

Wang, H.; Shi, X.; and Yeung, D.-Y. 2015. Relational stacked denoising autoencoder for tag recommendation. In *AAAI*, 3052–3058.

Wang, H.; Wang, N.; and Yeung, D.-Y. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1235–1244. ACM.