Signal Processing 169 (2020) 107366

Contents lists available at ScienceDirect

Signal Processing

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

Deep latent factor model for collaborative filtering

Aanchal Mongia [a], Neha Jhamb [a], Emilie Chouzenoux [b], Angshul Majumdar [a,*]

[a] Indraprastha Institute of Information Technology, New Delhi 110020, India
[b] Université Paris-Saclay, CentraleSupélec, Inria, CVN, 91190, Gif-sur-Yvette, France

ARTICLE INFO

Article history:
Received 16 May 2019
Revised 12 October 2019
Accepted 5 November 2019
Available online 11 November 2019

Keywords:
Deep learning
Latent semantic analysis
Collaborative filtering
Recommender systems

ABSTRACT

Latent factor models have been used widely in collaborative filtering based recommender systems. In recent years, deep learning has been successful in solving a wide variety of machine learning problems. Motivated by the success of deep learning, we propose a deeper version of latent factor model. Experiments on benchmark datasets shows that our proposed technique significantly outperforms all state-of-the-art collaborative filtering techniques.

1. Introduction

Today much of retail business is online. Retail ecommerce is having such an impact on economy that economists are dubbing it as the 'Amazon effect'. Ecommerce is generating more jobs and checking inflation rates (at least in the USA). Most of its success is owing to recommender systems.

Unlike physical stores, ecommerce portals deal with literally millions of products. It is not possible for the customer / user to sift through all the products in a finite amount of time to find out what he / she needs. The users rely on the suggestions from the recommender system to zero-in on products they like.

The benefit of recommender system goes in both ways. If the recommendations are not good, the user may stop relying on its suggestions and stop using the ecommerce portal. This harms the user – as he / she does not get what is required; it is also detrimental to the portal – as it loses out on revenue. To improve customer satisfaction, goodwill and revenue, it is thus mandatory for the ecommerce portals to have a very accurate recommendation system.

In the initial days of recommender systems, classical information retrieval based approaches such as content based filtering were used. However, such techniques relied on expert opinions and were very sector specific (fashion and books would be treated completely differently) and hard to generalize as algorithms. This led to its early demise. For the last two decades collaborative filtering has been the de facto approach behind recommender systems.

Collaborative filtering can be categorized into several branches. The initial approach was based on neighbourhood based models. They were intuitive, easy to understand and implement but lacked in accuracy. The second approach was based on classification. This yields slightly better results but is very hard to explain. The third, and the most prominent approach today is based on latent factor models. This class of techniques is more abstract and requires good understanding of mathematics for interpretation, but are nevertheless more accurate.

In the last half a decade deep learning has made inroads into almost all aspects of applied computer science – speech processing, computer vision, NLP etc. It has also seen some applications in information retrieval. Motivated by the success of deep learning, we propose a deep latent factor model. This would be a new deep learning tool, specifically tailored for collaborative filtering problems.

The shallow / standard latent factor model was based on the matrix factorization approach. The advent of deep learning, generalized matrix factorization to deeper versions. In this work, we follow the same idea and extend the latent factor model to a deeper version. However, it must be noted that the extension from deep matrix factorization techniques to our deep latent factor model is non-trivial. In the former, all the observations are known but, in our case, only a partial set of observations are known – this makes the problem more challenging.

* Corresponding author.
E-mail addresses: aanchalm@iiitd.ac.in (A. Mongia), neha16037@iiitd.ac.in (N. Jhamb), emilie.chouzenoux@centralesupelec.fr (E. Chouzenoux), angshul@iiitd.ac.in, angshulm@ece.ubc.ca (A. Majumdar).

https://doi.org/10.1016/j.sigpro.2019.107366
0165-1684/

2. Literature review

2.1. Neighborhood/similarity based models

Neighborhood based models try mimicking the nature of human interaction. It finds out users having similar taste as that of the active user. Among these similar users, weight is given according to the similarity; i.e. more the similarity more the weight. The rating on a particular item from the similar users is multiplied by these weights to estimate the rating from the active user.

To find out the active user's rating on a particular item, the ratings of the similar users are interpolated. This is given by,

$$v_{a,j} = \sum_{i \in neighbours(a)} w_i v_{i,j} \tag{1}$$

Here a is the active user; we want to predict the missing rating of user a for the item j. For this, we look at all the neighbours of a, indexed by i and interpolate their rating by multiplying it by the linear interpolation weights w_i. Such a technique is called user-user recommendation [1,2].

The way we have looked at collaborative filtering from the user's perspective, we can also look at it from the items perspective. The two are exactly similar. This leads to the item-item collaborative filtering [3,4]. Here instead of trying to find similar users one has to find similar items. The score of a user on the active item is found by interpolating the ratings given the same user on similar items. The interpolation weights, as previously are usually based on normalized similarity measure.

There are many variants to these basic approaches. Some studies combine the item and the user-based models [5]. Other studies have posed such similarity based collaborative filtering as a graph signal processing problem [6]; however the basic approach remains the same there in. Such techniques are simple to understand and analyze. They cannot compete with more recent abstract mathematical models but may be preferred by practitioners for the ease of implementation and understanding. Since neighborhood based models is not the topic of our interest, we do not discuss it any further. The interested reader can peruse [7].

2.2. Latent factor models

In content based filtering, one had to exclusively specify the factors that were thought responsible for user's choice on a particular class of items. As a result, the algorithms were not generalizable; the factors responsible for footwear were different from the ones responsible for selecting for instance movies. That is the primary reason for its failure.

Latent factor model [8-10] is a generalization of content based filtering. It is based on the assumption that the factors responsible for the user's choice on an item need not to be explicitly known. A user i can be defined by its affinity towards these latent factors, represented by u_i and an item j can be defined by its corresponding latent factor v_j. The rating is high when the two latent factors match (same as content based filtering). This is best modeled by the inner product between the user's and item's latent factors. The rating of the ith user on the jth item is modeled as:

$$x_{i,j} = u_i v_j, \forall i, j \tag{2}$$

Considering the entire ratings matrix for M users and N items, (2) can be represented as:

$$X = UV \text{where} U = [u_1|...|u_M] \text{and} V^T = [v_1|...|v_N]$$

Had the full ratings matrix be known, the problem would be trivial. What makes it challenging is the fact that the matrix is only partially observed; the goal is to infer the missing ratings. Once this is done, one can suggest items to users with high predicted ratings. Mathematically we can express it as:

$$Y = R \cdot X = R \cdot (UV) \tag{3}$$

Here R is a binary sampling mask consisting of 0 s where ratings are missing and 1 s where they are present; symbol $'\cdot'$ indicates element-wise product.

One can estimate the latent factor matrices for the users and the items by solving the following problem.

$$\min_{U,V} \|Y - R \cdot (UV)\|_F^2 + \lambda\left(\|U\|_F^2 + \|V\|_F^2\right) \tag{4}$$

where $\|.\|_F$ denotes the Frobenius norm. The ridge regression type penalties are used to overcome over-fitting. There are many algorithms for solving (4) starting from simple alternative least squares to multiplicative updates or block conjugate gradients. For all practical purposes, the simple alternating least squares usually yields good results.

The cost function involved in the factorization problem (4) is bi-linear in the variables U and V. Therefore, there is usually no guarantee of convergence of iterative solvers to a global minimum; guarantees typically hold only for convergence to local minima. This issue can be overcome by directly solving for the ratings themselves instead of the latent factors.

The assumption here is that the ratings matrix is of low-rank. This follows directly from the latent factor model; the rank of the matrix is the same as the number of factors. The most direct way to solve for the ratings would be to find an X explaining optimally the data Y, and with a minimal rank. But the rank minimization is known to be NP hard and hence there is no tractable solution (every algorithm is as good as brute force search). To alleviate this problem, theoretical studies [11-13] have shown that one can guarantee a low rank solution (under certain assumptions) by relaxing the NP hard rank minimization problem to its closest convex surrogate – relying on the nuclear norm. Mathematically this is expressed as,

$$\min_X \|Y - R \cdot X\|_F^2 + \lambda \|X\|_{NN} \tag{5}$$

Here $\|\cdot\|_{\text{NN}}$ denotes the nuclear norm; defined as the sum of singular values. Problem (5) is a convex problem that can be solved, for instance, by semi-definite programming solvers. Today more efficient algorithms exist [14].

The number of entries in Y are actually far fewer than the number of entries in X; for academic problems, only 5% of the data is available and in practical scenarios, less than 1% of the data is available. This makes collaborative filtering a highly underdetermined problem. In such a context, any secondary information is likely to improve the results. For example, studies like [15,16] have shown that using the associated metadata (user's demographic and item's metadata) can indeed improve recommendation accuracy. Other studies [17,18] showed that harnessing the power of neighborhood based models into the latent factor based method can also improve the results. In this work, we are not concerned about using associated information, and hence these techniques will not be discussed any further.

2.3. Representation learning

Restricted Boltzmann machine (RBM) [19,20] (Fig. 1) is popular today as a building block for deep belief network; but it was originally introduced for solving the collaborative filtering problem. However, owing to its inherent restrictions, foremost among them being the constraint on the input to be 1 or 0, RBMs never became popular in the context of collaborative filtering. Neither are the modified Gaussian Bernoulli RBMs, which expect continuous valued inputs in the range between 0 and 1, suitable for such inputs.

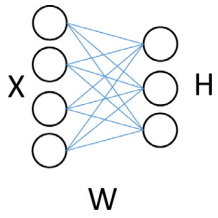

Fig. 1. Restricted Boltzmann machine.

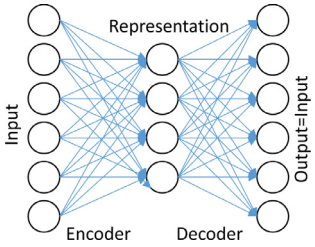

Fig. 2. Autoencoder.

Drawing similarity with the latent factor models, one can see that the network weights can be seen as users' latent factors and the representation as items' latent factors. The RBM is learnt by contrastive divergence. This is a cumbersome technique not amenable to mathematical manipulations.

In recent times, autoencoder based collaborative filtering techniques have shown promising results [21-26]. They rely on a typical neural network where the output is the same as the input. The schematic diagram is shown in Fig. 2. When used for collaborative filtering, the input (and output) has missing values. When there are such missing entries in the data, the corresponding network weights are not updated. Once the training is finished, the representation from the encoder is multiplied by the decoder to get the full ratings matrix.

Deep / stacked autoencoders have also been used for collaborative filtering. Conceptually, the input and the outputs remain the same. The only difference between the shallow (Fig. 2) and deep autoencoder is that the later has multiple layers of encoders and decoders. However, there is no significant gain in going deeper as was shown in [23,24].

In recent years, with the success of deep learning in almost all areas of applied machine learning, such techniques have been leveraged for collaborative filtering as well; see for instance [23-25].

Most other studies in deep learning based collaborative filtering are somewhat heuristic. For example, in [26], the inputs to the deep neural network are simply the IDs of the user and the item and the output is the corresponding rating. Such a model is likely to be arbitrary since the ID of the user and the item do not carry information about each other. Therefore, it makes limited sense to predict rating from such inputs.

The work [27] is worthy to be mentioned. Instead of using user's and item's IDs as inputs, it characterizes each user by his/her ratings on all items and characterizes each item by all its available ratings. This too uses a classification based framework where the output is the corresponding rating of the user on the item. Though a more meaningful approach than [26], using ratings for both inputs (user and item) and as the output (class) appears difficult to justify. Unsurprisingly, neither [26] nor [27] provides any justification for their model.

The most thorough and logical approach to deep learning based collaborative filtering has been proposed in [28]. It relies on a deep neural network based regression framework. It uses user's past history (of item ratings) as input and the top recommended new item as output. Note that this approach does not perform collaborative filtering in the conventional sense since it does not predict ratings.

There are two studies that proposed deep matrix factorization [29,30]. These studies factor a fully observed matrix into multiple matrices. They do not pertain to collaborative filtering where the task is to recover a partially observed matrix. Nevertheless, we mention these studies owing to their relationship with matrix factorization.

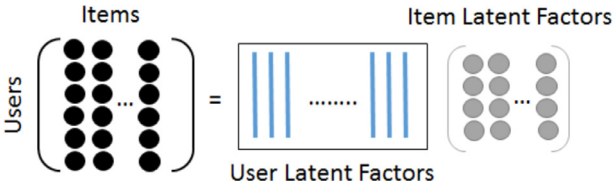

Fig. 3. Latent factor model.

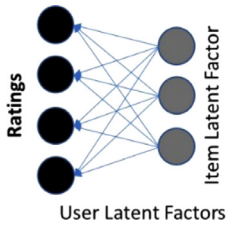

Fig. 4. Neural network interpretation.

3. Proposed deep latent factor model

3.1. Latent factor model as neural network

Fig. 3 depicts the standard latent factor model. We have assumed users to be along the rows and items along the columns. The entire ratings matrix is expressed as a product of user latent factor matrix and item latent factor matrix. Being a product of two matrices it is expressed mathematically in the form of matrix factorization. For the convenience of the reader, we repeat (2).

$$X = UV$$

where U denotes the user latent factors and V the item latent factors.

In this work we look at the latent factor model as a neural network. Instead of looking at the user latent factors as vectors, we can think of them as connections from the item latent factors to the ratings. This is shown in Fig. 4.

The input to this neural network are all the users' ratings on the jth item denoted by x_j (black nodes) in Fig. 4. Their corresponding representation is the latent factors for the corresponding item v_j (gray nodes). Following (2), the relationship between both is expressed as $x_j = Uv_j$ - this is same as the equation of a neural network, albeit in an opposite direction from the representation to the input. Each row u_i of U is the user latent factor for the ith user.

Once we have the neural network type interpretation of collaborative filtering, it is easy to conceptualize deeper extensions. In deep learning architectures, the latent representation from one layer acts as an input to the subsequent layer. We follow the same principle in proposing the deep latent factor model.

3.2. Deep latent factor model

A deeper (2-level) latent factor model is shown in Fig. 5. This can be extended to deeper layers. Like all other deep learning tools, we suffer from the limitations of mathematical interpretability. Intuitively speaking, as one goes deeper, more abstract representations are observed. For example, the latent factor model stems

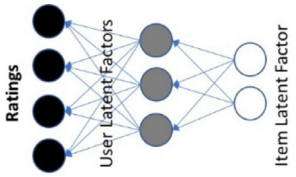

Fig. 5. Deep latent factor model.

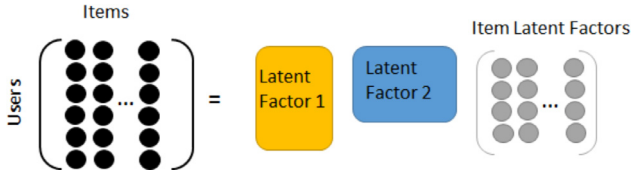

Fig. 6. Factorization type interpretation.

from the assumption that there are only a few factors that guide or decision in choosing a book or a movie. For a book, it may be just the name of the author, or the genre or the publisher. For movies the number of factors may be slightly more – it can be the star cast, the director, the main technical crew. It is not possible to capture all the factors objectively, which justifies the latent factor model. In a deeper latent factor model, instead of capturing the superficial factors (author, actor etc.) it is likely that deeper personality traits are captured. For example, the deeper nodes may refer to five classic personality traits [31] – *introversion, extroversion, neuroticism, openness* and *conscience*. Expressing the models based on such abstract fundamental traits is likely to make it more robust and accurate.

The neural network type interpretation is not mandatory, but it is easier to align with other deep learning methods. One can express the deep latent factor model in the more usual matrix factorization form (Fig. 6).

Mathematical formulation

In the standard latent factor model the ratings matrix is expressed as a product of user and item latent factors.

$$X = UV \tag{6}$$

Usually non-negativity is enforced on the latent factors, in a similar manner than in the rectified linear unit (ReLU) type activation function in neural network.

In our deeper model, we will have multiple levels of user latent factors and one final level of item latent factors. In the case of 2 layers, this can be expressed as,

$$X = U_1U_2V \tag{7}$$

Here U_1 and U_2 are the two levels of item latent factors and V consists of the deep user latent factors. Such a deep factorization model has been proposed before [29,30]. In each level, a ReLU type activation was imposed by non-negativity constraints. The formulation (7) can be easily extended to N layers:

$$X = U_1U_2...U_NV \tag{8}$$

Techniques for solving deep matrix factorization have already been developed [29,30] when the data is fully observed. Unfortunately, in collaborative filtering, the ratings matrix is only partially observed, i.e.:

$$Y = R \cdot X \tag{9}$$

where X is the full ratings matrix, R the binary matrix of 1′s and 0′s and Y the acquired ratings matrix.

We incorporate the deep matrix factorization framework (7) into the partially observed model (8) to yield our deep latent factor model.

$$Y = R \cdot X = R \cdot (U_1U_2...U_NV) \tag{10}$$

Solution

We will show the algorithm for three layers; it will be generic enough for more or fewer layers. The formulation for three layers is –

$$\begin{aligned} &\min_{U_1,U_2,U_3,V} \tfrac{1}{2}\|Y - R\cdot(U_1U_2U_3V)\|_F^2 \text{suchthat} \\ &U_1U_2U_3V \geq 0, U_1 \geq 0, U_2 \geq 0, U_3 \geq 0, V \geq 0 \end{aligned} \tag{11}$$

This is equivalent to the following,

$$\min_{U_1,U_2,U_3,V,X} \tfrac{1}{2}\|Y - R\cdot(U_1U_2U_3V)\|_F^2 \tag{12}$$

$$\text{suchthat} U_1U_2U_3V = X \text{and} X \geq 0, U_1 \geq 0, U_2 \geq 0, U_3 \geq 0, V \geq 0.$$

We propose a projected gradient method [32], with inner loop based on alternating projection [33] to solve (12). The general form of our algorithm is as follows:

$$\begin{aligned} &\text{Initialize} : X^0, U_1^0, U_2^0, U_3^0, V^0 \\ &\text{For} k = 1, 2, ... \\ &(1)\ \bar{X}^k = X^k - \gamma\left(R^T \cdot \left(R \cdot X^k - Y\right)\right) \\ &(2)\ \left(X^{k+1}, U_1^{k+1}, U_2^{k+1}, U_3^{k+1}, V^{k+1}\right) \text{solutionof} \\ &\min_{X,U_1,U_2,U_3,V} \left\|\bar{X}^k - X\right\|_F^2 \text{s.t.} U_1U_2U_3V = X \\ &\text{and} X \geq 0, U_1 \geq 0, U_2 \geq 0, U_3 \geq 0, V \geq 0 \end{aligned}$$

The second sub-problem corresponds to projecting X^k on the constrained domains $U_1U_2U_3V=X$ and $X \geq 0$, $U_1 \geq 0$, $U_2 \geq 0$, $U_3 \geq 0$, $V \geq 0$. This projection problem is complex, as the equality constraint defines a non-convex set. Moreover, it has no closed form. We thus propose to perform alternating projections on the constraints, to derive an approximate solution to this subproblem. Each variable will be treated sequentially, in a Gauss-Seidel fashion, and then projected on its associated constraint set. For variables U_i and V, we propose to approximate the projection of the intersection of positive and equality constraint, by the composition of the respective projectors, in order to avoid inner iterations.

Step 2 will read as follows,

$$\begin{aligned} X^{k+1} &= P_+\left(\bar{X}^k\right) \\ U_1^{k+1} &= P_+\left(P_{\left\{U_1U_2^kU_3^kV^k = X^{k+1}\right\}}\left(U_1^k\right)\right) \\ U_2^{k+1} &= P_+\left(P_{\left\{U_1^{k+1}U_2U_3^kV^k = X^{k+1}\right\}}\left(U_2^k\right)\right) \\ U_3^{k+1} &= P_+\left(P_{\left\{U_1^{k+1}U_2^{k+1}U_3V^k = X^{k+1}\right\}}\left(U_3^k\right)\right) \\ V^{k+1} &= P_+\left(P_{\left\{U_1^{k+1}U_2^{k+1}U_3^{k+1}V = X^{k+1}\right\}}\left(V^k\right)\right) \end{aligned}$$

Hereabove, P_+ denotes the projector onto the positive orthant, i.e. capping the negative entries of an input matrix to 0. We can then apply the general property that the projection of a matrix U on a linear constraint $X = AUB$ is given by

$$\hat{U} = U - A^\dagger(AUB - X)B^\dagger \tag{13}$$

with † the pseudo-inverse operation, so that step 2 finally reads:

$$\begin{aligned} X^{k+1} &= P_+\left(\bar{X}^k\right) \\ U_1^{k+1} &= P_+\left(U_1^k - \left(U_1^kU_2^kU_3^kV^k - X^{k+1}\right)\left(U_2^kU_3^kV^k\right)^\dagger\right) \\ U_2^{k+1} &= P_+\left(U_2^k - \left(U_1^{k+1}\right)^\dagger\left(U_1^{k+1}U_2^kU_3^kV^k - X^{k+1}\right)\left(U_3^kV^k\right)^\dagger\right) \\ U_3^{k+1} &= P_+\left(U_3^k - \left(U_1^{k+1}U_2^{k+1}\right)^\dagger\left(U_1^{k+1}U_2^{k+1}U_3^kV^k - X^{k+1}\right)\left(V^k\right)^\dagger\right) \\ V^{k+1} &= P_+\left(V^k - \left(U_1^{k+1}U_2^{k+1}U_3^{k+1}\right)^\dagger\left(U_1^{k+1}U_2^{k+1}U_3^{k+1}V^k - X^{k+1}\right)\right) \end{aligned}$$

Steps 1 and 2 can be understood as a gradient projection method. The complicated form of the constraint requires the use

of an inner step for approximating the projection step. The convergence of the whole scheme cannot be established easily, because of the presence of the coupling in the equality constraint $U_1U_2U_3V = X$. However, in our experimental results, we will show that the algorithm converges empirically. Here we have shown the algorithm for three layers. This can be generalized to any number of layers. The general algorithm is given in the appendix.

A recent work [34], proposes a hierarchical latent factor model. Unlike ours, which just goes deep, [34] bifurcates in each layer. The resulting model increases complexity since there are more variables to solve now and extra constraints to impose the tree structure. This is likely to make the model more vulnerable to overfitting. The work claims convergence, but they provide no insights into this claim.

Computational complexity

The computational cost of the proposed method is governed by the pseudo-inverse operations that must be performed in Step 2. The complexity of it is given by $O(n^w)$ where $w<2.37$ and is conjectured to be 2. Note however that one can play with the size of the latent factor matrices, in order to control the memory and computational burden of these inversions. An alternative for very large datasets is to use conjugate gradient solver, but in our practical experiments, it appears not necessary.

4. Experimental evaluation

4.1. Datasets

We carry our evaluation on movie recommendations. Experiments are carried out on three standard datasets – Movielens 100 K, Movielens 1 M, Movielens and 10 M. All of them are from https://grouplens.org/datasets/movielens/.

(1) movie-100K: 100,000 ratings for 1682 movies by 943 users;
(2) movie-1M: 1 million ratings for 3900 movies by 6040 users;
(3) movie-10M: 10 million ratings for 10,681 movies by 71,567 users.

For these datasets the splits between training and test sets are already pre-defined. The protocol is to carry out 5 fold cross-validation on these sets.

4.2. Convergence

We show the empirical convergence of our algorithm on the 100 K. One can see that our algorithm decreases the cost function monotonically. The results from the other two datasets are similar and are not shown here (Fig. 6a).

4.3. Comparative results

We have compared our technique with some state-of-the-art deep methods – collaborative deep learning (CDL) [25], marginalized deep autoencoder (MDA) [24] and deep matrix factorization (DMF) [28]. We also compare with a recent shallow yet robust approach called robust matrix factorization (RMF) [35]; although it has never been used for collaborative filtering, it surpasses standard matrix factorization for other tasks. We have finally compared with the hierarchical latent factor model (HLFM) [34].

For our proposed method we need to specify three parameters – 1. The number of layers, the number of basis in each layer and the value of γ. Usually the number of layers is determined by the number of samples in the database. Since our datasets have less than 10,000 samples (users / items) it is unlikely that our architecture can go beyond 3 layers and produce good results; this is usually a rule of thumb.

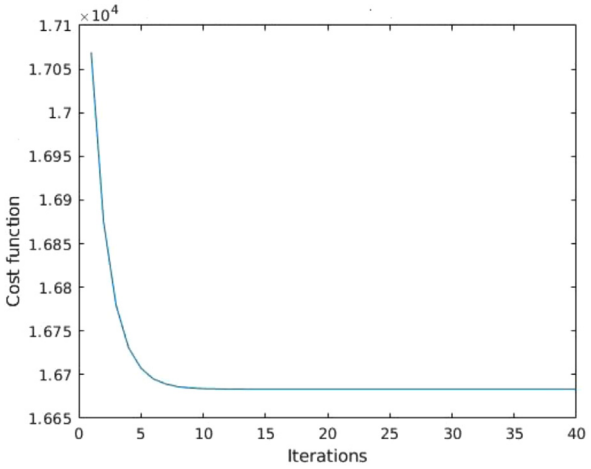

Fig. 6a. Empirical convergence plot.

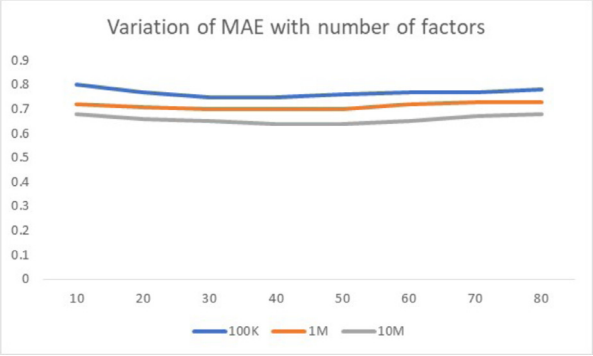

Fig. 7. MAE vs Number of factors for matrix factorization.

Following deep learning literature, we can fix the number of factors in each layer by halving in subsequent layers. However, we need to find the number of factors in the first layer. This, we do empirically by using a single layer matrix factorization. The number of latent factors are varied from 10 to 80 in steps of 10. The results are shown in Fig. 7. We can see that the best results are obtained for 40 latent factors. This determines the number of latent factors for the two and three layer model. The two levels architecture is 40-20 and the three levels architecture is 40-20-10. We have also shown results for a 4 layer architecture (40-20-10-5); this was mainly done to show that the results deteriorate after the third layer.

Finally, we need to specify γ, that is the stepsize in our projected gradient method; we have used $\gamma = 0.1$ throughout. This value was obtained by 4 fold cross-validation on the training set. We also found that our algorithm is robust to the choice of parameter between 0.01 to 0.8. Outside the said range, the performance degrades gracefully.

For initialization, we make use of the following deterministic SVD based strategy, that appears to lead to stable and accurate results. The missing entries are first filled by the row and column averages. SVD is performed on the thus filled matrix. The left singular vectors are used for initializing the U_1. Then SVD is performed on the remaining portion, i.e. the product of singular values and the right singular vectors. The left singular vectors of the second SVD is used to initialize U_2. This process is continued. For the final layer, the product of the singular values and the right singular vectors is used to initialize V.

One must note that the methods compared against (MDA, CDL, DMF, RMF and HLFM) have not been used on standard protocols

Table 1
Results on 100 K Movielens.

Method	MAE	RMSE	Precision		Recall	
			@10	@20	@10	@20
RMF	.732	.938	.515	.380	.641	.767
DMF	.735	.940	.513	.380	.641	.775
CDL	.742	.953	.510	.377	.642	.765
MDA	.758	.981	.513	.372	.641	.767
HLFM	.750	.962	.512	.376	.640	.769
40-20	**.726**	**.939**	**.522**	**.380**	**.649**	**.782**
40-20-10	**.717**	**.901**	**.536**	**.399**	**.658**	**.792**
40-20-10-5	**.732**	**.941**	**.520**	**.376**	**.642**	**.778**

Table 2
Results on 1 M Movielens.

Method	MAE	RMSE	Precision		Recall	
			@10	@20	@10	@20
RMF	.689	.876	.669	.526	.625	.792
DMF	.691	.878	.671	.523	.625	.799
CDL	.689	.871	.671	.531	.630	.802
MDA	.686	.879	.669	.526	.625	.791
HLFM	.698	.880	.661	.517	.619	.785
40-20	**.681**	**.864**	**.673**	**.531**	**.636**	**.799**
40-20-10	**.678**	**.854**	**.691**	**.543**	**.641**	**.809**
40-20-10-5	**.682**	**.866**	**.670**	**.528**	**.635**	**.796**

Table 3
Results on 10 M Movielens.

Method	MAE	RMSE	Precision		Recall	
			@10	@20	@10	@20
RMF	.630	.810	.682	.559	.629	.803
DMF	.618	.805	.671	.569	.631	.810
CDL	.616	.802	.672	.568	.633	.810
MDA	.621	.816	.680	.555	.620	.802
HLFM	Does not run at this scale					
40-20	**.613**	**.802**	**.689**	**.569**	**.632**	**.813**
40-20-10	**.600**	**.794**	**.696**	**.579**	**.640**	**.820**
40-20-10-5	**.608**	**.798**	**.693**	**.572**	**.638**	**.815**

defined on all these datasets. Therefore, we have followed the approach presented in the corresponding papers to tune the parameters for our datasets.

As the evaluation metric, we show results on all the standard ones – mean absolute error (MAE), root mean squared error (RMSE), precision and recall. The results are shown in the following Tables 1-3. The results from our method are shown in bold. We see that, the results improve from 2 to 3 layers; but it degrades from 3 to 4 layers. This is because in deep learning it is believed that going deeper improves abstraction capacity which in turn boosts results. But one cannot go on going deeper, as it requires learning more parameters. In such limited data scenario, the need to learn more parameters leads to overfitting; this reduces the performance if one goes too deep.

One can see that we improve upon the rest in terms of every possible metric; the improvements are considerably large. To put the results in perspective, one must remember that the Netflix prize of 1 million was given to the winners who reduced the RMSE on the Netflix dataset from 0.95 to 0.85. We do not give results on the Netflix dataset owing to concerns over a pending lawsuit on the usage of the dataset.

5. Conclusion

This work introduces the deep latent factor model (deepLFM). We have compared with other deep and shallow techniques and shown that the proposed method outperforms all; at least on the benchmark databases compared on.

At a later stage we would like to incorporate neighborhood information from the users and items in a graph based deep latent factor framework. This will borrow ideas from graph signal processing [36,37]. The basic idea would be to regularize the proposed model with trace of graph Laplacians in a fashion like graph regularized matrix factorization [37]. These graph Laplacians will be defined from the similarities of the users and the items.

In recent years, tensor decomposition / factorization strategies have been gaining importance in signal processing and machine learning [38,39]. The way we do multi-level factorizations in this work, it would be interesting to introduce multi-level tensor factorization in the future.

Collaborative filtering / recommender systems have benefited from the use of additional user-item metadata that has been incorporated into the matrix completion framework via graph regularization [40,41]. In the future, we would like to improve our method in a similar fashion using graph regularization.

Declaration of Competing Interest

None.

Acknowledgement

This work is supported by the Infosys Center for Artificial Intelligence @ IIIT Delhi and by the Indo-French CEFIPRA grant DST-CNRS-2016-02.

Appendix

Incorporating arbitrary constraints

Although we are unaware of any other constraint / that is applied for latent factor model other than simple positivity constraints, the reader may have an interest to modify our proposed model to other problems. We show that it would actually be easy to incorporate any type of constraint into our proposed framework. The complete mathematical formulation is given in (11). If one wants to add some regularization, one simply needs to add them to (11).

$$\min_{U_1,U_2,U_3,V} \frac{1}{2}\|Y - R \cdot (U_1U_2U_3V)\|_F^2 \text{ such that}$$

$$U_1U_2U_3V \geq 0, U_1 \in C_1, U_2 \in C_2, U_3 \in C_3, V \in S$$

with C_1, C_2, C_3 and S some closed convex sets incorporating both positivity and regularity constraints on the factors U_1, U_2, U_3 and V, respectively. The alternating projection algorithm would remain almost unchanged. The only difference is that the projection steps must now be performed on the sets C_1, C_2, C_3 and S, instead of on the positive orthant.

$$X^{k+1} = P_+\left(\bar{X}^k\right)$$

$$U_1^{k+1} = P_{C_1}\left(U_1^k - \left(U_1^kU_2^kU_3^kV^k - X^{k+1}\right)\left(U_2^kU_3^kV^k\right)^\dagger\right)$$

$$U_2^{k+1} = P_{C_2}\left(U_2^k - \left(U_1^{k+1}\right)^\dagger\left(U_1^{k+1}U_2^kU_3^kV^k - X^{k+1}\right)\left(U_3^kV^k\right)^\dagger\right)$$

$$U_3^{k+1} = P_{C_3}\left(U_3^k - \left(U_1^{k+1}U_2^{k+1}\right)^\dagger\left(U_1^{k+1}U_2^{k+1}U_3^kV^k - X^{k+1}\right)\left(V^k\right)^\dagger\right)$$

$$V^{k+1} = P_S\left(V^k - \left(U_1^{k+1}U_2^{k+1}U_3^{k+1}\right)^\dagger\left(U_1^{k+1}U_2^{k+1}U_3^{k+1}V^k - X^{k+1}\right)\right)$$

For example, consider, for $i = 1, 2, 3$,

$$C_i = \{U_i \geq 0 \text{ and } ||U_i||_F \leq \theta_i\}$$

with θ_i positive regularization factors constraining the l2-norms on U_i, and

$$S = \{V \geq 0 \text{ and } ||V||_1 \leq \mu\}$$

with μ a positive bound on the sought l_1-norm on V. It is worth pointing out that in that case, the projectors remain with a closed form. First, for every i we have, as a consequence of Combettes [42] (Remark 3.15):

$$P_{C_i}(U_i) = \begin{cases} \frac{\theta_i}{||P_+(U_i)||_F} P_+(U_i) & \text{if } ||P_+(U_i)||_F \rangle \theta_i \\ P_+(U_i) & \text{elsewhere} \end{cases}$$

Moreover,

$$P_S(V) = \begin{cases} P_+(V) & \text{if } ||V||_1 \leq \mu \\ P_{simplex}(V) & \text{elsewhere} \end{cases}$$

with $P_{simplex}(V)$ the projection on simplex ball with radius μ, calculated for instance using the fast algorithm from Condat [43].

Extension for n-layer latent factor model

For an n-layer deep latent factor model, we need to solve the following generalized version of (11),

$$\min_{U_1, U_2, \ldots, U_N, V} \frac{1}{2} \|Y - R \cdot (U_1 U_2 \ldots U_N V)\|_F^2 \text{suchthat} U_1 U_2 \ldots U_N V \geq 0,$$
$$U_i \geq 0 i = 1 \ldots N, V \geq 0$$

One can see that the outer loop is independent of the depth of the model. When going from three to a higher number of layers, the main change lies in adding steps in the alternating projection inner loop, in order to take into account the update of the layers. The updates for each variable would read:

$$X^{k+1} = P_+\left(\bar{X}^k\right)$$

$$U_1^{k+1} = P_+\left(U_1^k - \left(U_1^k U_2^k U_3^k .. U_N^k V^k - X^{k+1}\right)\left(U_2^k U_3^k .. U_N^k V^k\right)^\dagger\right)$$

$$U_2^{k+1} = P_+\left(U_2^k - \left(U_1^{k+1}\right)^\dagger \left(U_1^{k+1} U_2^k U_3^k .. U_N^k V^k - X^{k+1}\right)\left(U_3^k .. U_N^k V^k\right)^\dagger\right)$$

$$U_3^{k+1} = P_+\left(U_3^k - \left(U_1^{k+1} U_2^{k+1}\right)^\dagger \left(U_1^{k+1} U_2^{k+1} U_3^k .. U_N^k V^k - X^{k+1}\right)\left(U_4^k .. U_N^k V^k\right)^\dagger\right)$$

$$U_N^{k+1} = P_+\left(U_N^k - \left(U_1^{k+1} U_2^{k+1} .. U_{N-1}^{k+1}\right)^\dagger \left(U_1^{k+1} U_2^{k+1} .. U_{N-1}^{k+1} U_N^k V^k - X^{k+1}\right)\left(V^k\right)^\dagger\right)$$

$$V^{k+1} = P_+\left(V^k - \left(U_1^{k+1} U_2^{k+1} U_3^{k+1} \ldots U_N^{k+1}\right)^\dagger \left(U_1^{k+1} U_2^{k+1} U_3^{k+1} \ldots U_N^{k+1} V^k - X^{k+1}\right)\right)$$

References

[1] J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, ACM SIGIR Forum 51 (2) (2017) 227–234.
[2] J.S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc, 1998, pp. 43–52.
[3] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collabo-rative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, ACM, 2001, pp. 285–295.
[4] G. Linden, B. Smith, J. York, Amazon. com recommendations: item-to-item collaborative filtering, IEEE Internet Comput. (1) (2003) 76–80.
[5] J. Wang, A.P. De Vries, M.J. Reinders, Unifying user-based and item-based collaborative filtering approaches by similarity fusion, in: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2006, pp. 501–508.
[6] W. Huang, A.G. Marques, A.R. Ribeiro, Rating prediction via graph signal processing, IEEE Trans. Signal Process. 66 (19) (2018) 5066–5081.
[7] J. Bobadilla, F. Ortega, A. Hernando, A. Gutierrez, Recommender systems survey, Knowl. Based Syst. 46 (2013) 109–132.
[8] T. Hofmann, Latent semantic models for collaborative filtering, ACM Trans. Inf. Syst. (TOIS) 22 (1) (2004) 89–115.
[9] T. Hofmann, J. Puzicha, Latent class models for collaborative filtering, IJCAI, 99, 1999.
[10] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 8 (2009) 30–37.
[11] E.J. Candes', T. Tao, The power of convex relaxation: near-optimal matrix completion, IEEE Trans. Inf. Theory 56 (5) (2010) 2053–2080.
[12] B. Recht, A simpler approach to matrix completion, J. Mach. Learn. Res. 12 (Dec) (2011) 3413–3430.
[13] X. Peng, C. Lu, Z. Yi, H. Tang, Connections between nuclear-norm and frobenius-norm-based representations, IEEE Trans. Neural Netw. Learn. Syst. 29 (1) (2018) 218–224.
[14] A. Majumdar, R.K. Ward, Some empirical advances in matrix completion, Signal Process. 91 (5) (2011) 1334–1338.
[15] A. Gogna, A. Majumdar, Matrix completion incorporating auxiliary information for recommender system design, Expert Syst. Appl. 42 (14) (2015) 5789–5799.
[16] F. Zhao, M. Xiao, Y. Guo, Predictive collaborative filtering with side information, in: IJCAI, 2016, pp. 2385–2391.
[17] Q. Gu, J. Zhou, C. Ding, Collaborative filtering: weighted non-negative matrix factorization incorporating user and item graphs, in: Proceedings of the 2010 SIAM International Conference on Data Mining, SIAM, 2010, pp. 199–210.
[18] N. Rao, H.-F. Yu, P.K. Ravikumar, I.S. Dhillon, Collaborative filtering with graph information: consistency and scalable methods, in: Advances in Neural Information Processing Systems, 2015, pp. 2107–2115.
[19] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted boltzmann machines for collaborative filtering, in: Proceedings of the 24th International Conference on Machine Learning, ACM, 2007, pp. 791–798.
[20] K. Georgiev, P. Nakov, A non-iid framework for collaborative filtering with restricted boltzmann machines, in: International Conference on Machine Learning, 2013, pp. 1148–1156.
[21] S. Sedhain, A.K. Menon, S. Sanner, L. Xie, Autorec: autoencoders meet collaborative filtering, in: Proceedings of the 24th International Conference on World Wide Web, ACM, 2015, pp. 111–112.
[22] Y. Ouyang, W. Liu, W. Rong, Z. Xiong, Autoencoder-based collaborative filtering, in: International Conference on Neural Information Processing, Springer, 2014, pp. 284–291.
[23] Y. Wu, C. DuBois, A.X. Zheng, M. Ester, Collaborative denoising auto-encoders for top-n recommender systems, in: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, ACM, 2016, pp. 153–162.
[24] S. Li, J. Kawale, Y. Fu, Deep collaborative filtering via marginal-ized denoising auto-encoder, in: Proceedings of the 24th ACM Inter-national on Conference on Information and Knowledge Management, ACM, 2015, pp. 811–820.
[25] H. Wang, N. Wang, D.-Y. Yeung, Collaborative deep learning for recommender systems, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 1235–1244.
[26] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, F. Zhang, A hybrid collaborative filtering model with deep structure for recommender systems, in: AAAI, 2017, pp. 1309–1315.
[27] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.
[28] H.-J. Xue, X. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, in: IJCAI, 2017, pp. 3203–3209.
[29] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, B.W. Schuller, A deep matrix factorization method for learning attribute representations, IEEE Trans. Pattern Anal. Mach. Intell. 39 (3) (2017) 417–429.
[30] Z. Li, J. Tang, Weakly supervised deep matrix factorization for social image understanding, IEEE Trans. Image Process. 26 (1) (2017) 276–288.
[31] L.A. Pervin, A critical analysis of current trait theory, Psychol. Inq. 5 (2) (1994) 103–113.
[32] W.W. Hager, H. Zhang, Recent advances in bound constrained optimization, in: F. Ceragioli, A. Dontchev, H. Futura, K. Marti, L. Pandolfi (Eds.), System Modeling and Optimization. CSMO 2005. IFIP International Federation for Information Processing, 199, Springer, Boston, MA, 2006.
[33] H.H. Bauschke, P.L. Combettes, Convex Analysis and Monotone Operator Theory in Hilbert Spaces, second ed., Springer, New York, 2017.
[34] S. Wang, J. Tang, Y. Wang, H. Liu, Exploring hierarchical structures for recommender systems, IEEE Trans. Knowl. Data Eng. 30 (6) (1 June 2018) 1022–1035.
[35] T. Liu, D. Tao, On the performance of manhattan nonnegative matrix factorization, IEEE Trans. Neural Netw. Learn. Syst. 27 (9) (Sept. 2016) 1851–1863.
[36] V. Borodin, H. Snoussi, F. Hnaien, N. Labadie, Signal processing on graphs: case of sampling in Paley–Wiener spaces, Signal Process. 152 (2018) 130–140.
[37] Q. Liao, Q. Zhang, Local coordinate based graph-regularized NMF for image representation, Signal Process. 124 (2016) 103–114.
[38] N.D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E.E. Papalexakis, C. Faloutsos, Tensor decomposition for signal processing and machine learning, IEEE Trans. Signal Process. 65 (13) (2017) 3551–3582 1 July1.
[39] T. Yokota, R. Zdunek, A. Cichocki, Y. Yamashita, Smooth nonnegative matrix and tensor factorizations for robust multi-way data analysis, Signal Process. 113 (2015) 234–249.

[40] K. Benzi, V. Kalofolias, X. Bresson, P. Vandergheynst, Song recommendation with non-negative matrix factorization and graph total variation, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 2439–2443.
[41] F. Monti, M.M. Bronstein, X. Bresson, Deep geometric matrix completion: a new way for recommender systems, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 6852–6856.
[42] P.L. Combettes, Monotone operator theory in convex optimization, Math. Program. Ser. A 170 (1) (2018) 177–206.
[43] L. Condat, Fast projection onto the simplex and the $\ell 1$ ball, Math. Program. Ser. A 158 (1–2) (2016) 575–585.