Neural Factorization Applied to Interaction Matrix for Recommendation

Ioannis Sarridis
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
isarridis@csd.auth.gr

Constantine Kotropoulos
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
costas@csd.auth.gr

***Abstract*—Matrix factorization methods have successfully been used for rating prediction. The interactions between users and items are recorded in the interaction matrix. In this paper, a neural matrix factorization method is proposed that is applied to the interaction matrix. More specifically, the normalized interaction matrix is given as input to the neural network in order to extract user and item embeddings. The estimated ratings are obtained by the inner product between the extracted user and item embeddings. The proposed method is assessed in movie rating prediction by employing three MovieLens datasets. It is demonstrated that the proposed neural factorization attains competitive performance and is less computationally demanding against the state-of-the-art methods in movie rating prediction.**

***Index Terms*—rating prediction, neural recommender system, matrix factorization**

I. Introduction

Nowadays, users are overloaded with a huge volume of information that does not necessarily match their interests. To improve user experience, many online services need to provide personalized information to users by employing recommender systems. For example, e-commerce businesses need to make personal recommendations to users about their products. Another example is movie recommendation that has attracted the interest of both academia and industry.

The underlying idea of a personalized recommender system is to model users' preferences on items based on their past activity (e.g., submitted ratings). This is known as Collaborative Filtering (CF). Matrix Factorization (MF) [1]–[3] is exploited into the most popular CF algorithms. MF models a user-item relation by projecting user and item vectors onto a latent space and computing their inner product. Over the last years, the research effort has led to efficient MF methods. A learning MF method was proposed in [4], coined as Funk Singular Value Decomposition (SVD). Funk SVD employs stochastic gradient descent optimization to learn user and item representations in the latent space. Unlike Funk SVD, SVD++ [5] employed user and item biases in order to learn the variation of rating values (i.e., independent of user-item interactions). An enhanced SVD++ method was developed in [5], which handles efficiently new users. It is known as Asymmetric SVD. Recently, many deep learning approaches to MF have been proposed, achieving competitive performance [6]–[8]. A neural autoregressive approach to collaborative filtering was proposed in [9]. It was coined as CF-NADE. It is based on Neural Autoregressive Distribution Estimator (NADE) [10] and Restricted Boltzmann Machine based Collaborative Filtering (RBM-CF) [11]. NADE is a distribution estimator applied to high dimensional binary vectors. RBM-CF is a generative graph model that consists of two layers. It aims to model the distributions of tabular data by generalizing Restricted Boltzmann Machines (RBMs). Inspired by RBM-CF, CF-NADE adapts NADE to CF tasks.

MF can be treated as a graph learning problem. In 2016, Kipf introduced Graph Convolutional Networks (GCN) [12]. One year later, a new approach of MF based on GCN was proposed in [13], which is known as Graph Convolutional Matrix Completion (GC-MC). GC-MC turns MF into a graph learning problem by employing a bipartite graph that consists of user-item interactions. Variational Graph Auto-Encoders were proposed in order to predict the links between user and items [14]. The Separable Recurrent Multi-graph CNN (sRMGCNN) that comprises graph convolutions and a Recurrent Neural Network, was proposed in [15]. sRMGCNN utilized 2 graphs to learn user/item embeddings, which are employed to reconstruct a user-item matrix.

In this paper, we propose a novel method, coined as Neural Interaction Matrix Factorization (NIMF). Interaction is defined as a user's activity related to an item (e.g., a user rates a movie, a customer likes a restaurant). The underlying idea of NIMF is to extract user and item neural representations, known as embeddings, in a latent space using the recorded interactions. NIMF utilizes the interaction matrix to learn the users/items embeddings, while [15] employs 2 different graphs (1 for users and 1 for items) for the same purpose. This is the major novelty introduced, which reduces significantly the time requirements of the proposed method against the top performing state-of-the-art ones. The inner product of the neural embeddings yields the estimated ratings. NIMF is tested on movie rating prediction. The target matrix comprises the estimated ratings. NIFM is tested on 3 movie datasets, namely MovieLens-100k, MovieLens-1m, MovieLens-10m, which consist of $100,000$,

Acknowledgement: This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH - CREATE - INNOVATE (project code: T1EDK-02474).

$1,000,000$, and $10,000,000$ ratings, respectively [16]. The objective figure of merit is the Root Mean Square Error (RMSE) between the estimated ratings and the corresponding actual (real) ratings. NIMF demonstrates competitive performance against state-of-the-art methods. Moreover, NIMF possesses less training parameters than its competitors, being less computationally demanding. The proposed method and classic MF methods share the same application fields, such as recommender systems, text mining, bioinformatics, etc.

The remainder of the paper is as follows. Section II describes briefly MF. Section III analyzes the proposed NIMF method. Section IV presents the experimental results. Section V concludes the paper and indicates future research directions.

II. MATRIX FACTORIZATION

MF techniques aim to project users and items in a latent space. Given a set of users, $U = \{u_1, u_2, \cdots, u_{|U|}\}$, and a set of items, $I = \{i_1, i_2, \cdots, i_{|I|}\}$, the rating matrix is defined as $\mathbf{R} = [R_{ui}]$. Whenever user u has rated item i, R_{ui} is non-zero. Otherwise, $R_{ui} = 0$. Let K be the number of latent factors. Many or few latent factors can lead to overfitting or underfitting, respectively. The user and item matrices in latent space are denoted as $\mathbf{U} \in \mathbb{R}^{|U| \times K}$ and $\mathbf{I} \in \mathbb{R}^{|I| \times K}$, respectively. The jth row of matrix $\mathbf{U}$ represents user u_j in the latent space and the ℓth row of matrix $\mathbf{I}$ represents item i_ℓ in the same latent space. The estimated rating matrix can be obtained by the factorization

$$\hat{\mathbf{R}} = \mathbf{U}\mathbf{I}^\top. \tag{1}$$

Accordingly, $\mathbf{U}$ and $\mathbf{I}$ have to be learned, so that a proper loss function is minimized. Several loss functions are employed in different MF methods. Here, the loss function is given by

$$\mathcal{L} = \frac{\|\hat{\mathbf{R}} - \mathbf{R}\|_F}{\sqrt{|U||I|}} + \lambda(\|\mathbf{U}\|_F^2 + \|\mathbf{I}\|_F^2), \tag{2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of the matrix argument and λ is a regularization parameter. The first term in Eq. (2) is the RMSE. Minimizing RMSE means that the estimated ratings become similar to real (actual) ratings. The second term in Eq. (2) introduces regularization, excluding extreme values in matrices $\mathbf{U}$ and $\mathbf{I}$ and prevents overfitting.

III. NEURAL INTERACTION MATRIX FACTORIZATION

Interactions between users and items contain crucial information about users' preferences. For example, let interactions be the users' ratings to movies. A user who rates only a specific movie genre implies that he or she prefers this genre. This kind of information is captured by the interaction matrix $\mathbf{A} \in \mathbb{R}^{|U| \times |I|}$ having elements:

$$A_{ui} = \begin{cases} 1, & \text{if } R_{ui} \neq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The interaction matrix $\mathbf{A}$ is also known as the bi-adjacency of the bipartite graph with parts U and I. The basic idea of NIMF

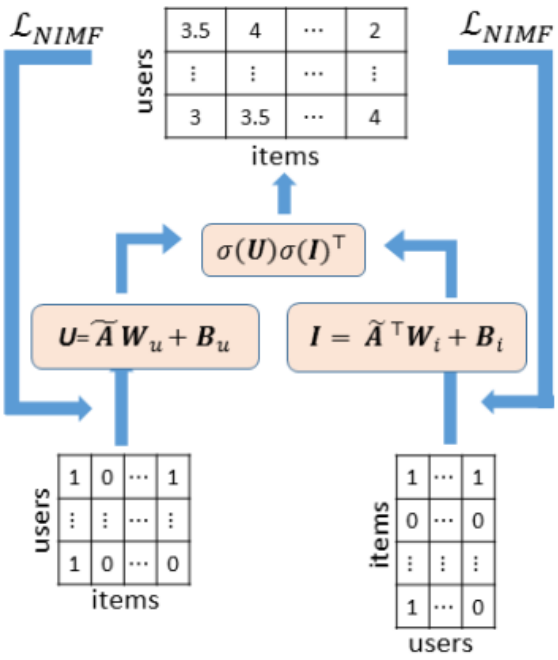

Fig. 1. The architecture of the proposed method NIMF.

is to learn matrices $\mathbf{U}$, $\mathbf{I}$ utilizing the interactions between users and items. Initially, matrix $\mathbf{A}$ has to be normalized. Let us define the diagonal matrix $\mathbf{D}_c \in \mathbb{R}^{|I| \times |I|}$ with elements:

$$\delta(i) = \sum_{u \in U} A_{ui},\ i \in I \tag{4}$$

and the diagonal matrix $\mathbf{D}_r \in \mathbb{R}^{|U| \times |U|}$ with elements:

$$\delta(u) = \sum_{i \in I} A_{ui},\ u \in U. \tag{5}$$

Next, the normalized interaction matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{|U| \times |I|}$ is given by:

$$\tilde{\mathbf{A}} = \mathbf{D}_r^{-1/2} \mathbf{A} \mathbf{D}_c^{-1/2}. \tag{6}$$

Inspired by the GCN layer [12], that performs a multiplication between an adjacency matrix and a weight matrix, here, the adjacency matrix is replaced by the interaction matrix. Let $\mathbf{B}_u = \mathbf{1}_{|U|}\, \mathbf{b}^\top \in \mathbb{R}^{|U| \times K}$ denote the bias matrix, where $\mathbf{1}_{|U|}$ is a vector of $|U|$ ones and $\mathbf{b} \in \mathbb{R}^{K \times 1}$ is the typical bias vector. Accordingly, the output of layer $\mathbf{U} \in \mathbb{R}^{|U| \times K}$ is

$$\mathbf{U} = \tilde{\mathbf{A}}\mathbf{W}_u + \mathbf{B}_u = \tilde{\mathbf{A}}\mathbf{W}_u + \mathbf{1}_{|U|}\, \mathbf{b}^\top, \tag{7}$$

where $\mathbf{W}_u \in \mathbb{R}^{|I| \times K}$ is the user weight matrix. The training parameters are the elements of $\mathbf{W}_u$. Similarly, matrix $\mathbf{I} \in \mathbb{R}^{|I| \times K}$ is obtained by

$$\mathbf{I} = \tilde{\mathbf{A}}^\top \mathbf{W}_i + \mathbf{B}_i = \tilde{\mathbf{A}}^\top \mathbf{W}_i + \mathbf{1}_{|I|}\, \mathbf{b}^\top, \tag{8}$$

where and $\mathbf{W}_i \in \mathbf{R}^{|U| \times K}$ is the item weight matrix. It can be shown that Eqs. (7) and (8) result by GCN [12] employing the adjacency matrix of the bipartite graph. NIMF network is

defined as the factorization between user embeddings $\sigma(\mathbf{U})$ and item embeddings $\sigma(\mathbf{I})$, i.e.,

$$\hat{\mathbf{R}} = f(\tilde{\mathbf{A}}) = \sigma(\mathbf{U})\sigma(\mathbf{I})^\top, \tag{9}$$

where σ is the activation function. Clearly, the elements of $\hat{\mathbf{R}}$ are inner products of user and item embeddings. The loss function is similar to Eq. (2). The only difference is that the regularization is applied to matrices $\mathbf{W}_u$ and $\mathbf{W}_i$ instead of $\mathbf{U}$ and $\mathbf{I}$. That is,

$$\mathcal{L}_{\text{NIMF}} = \frac{\|\hat{\mathbf{R}} - \mathbf{R}\|_F}{\sqrt{|U||I|}} + \lambda(\|\mathbf{W}_u\|_F^2 + \|\mathbf{W}_i\|_F^2). \tag{10}$$

The structure of NIMF is depicted in Fig. 1. Input is $\mathbf{A}$ (and $\mathbf{A}^\top$). There are two layers. The first layer passes the output of layer $\mathbf{U}$ through RELU, yielding $\sigma(\mathbf{U})$. Similarly, the second layer passes the output of layer $\mathbf{I}$ through RELU, yielding $\sigma(\mathbf{I})$. Network weights $\mathbf{W}_u$ and $\mathbf{W}_i$ are jointly optimized by minimizing Eq. (10) with stochastic gradient techniques. A neural network architecture mapping the row and column vectors of the interaction matrix into a latent space employing multiple convolutions has been proposed in [**?**]. Although the latter architecture shares some similarity with that depicted in Fig. 1, the proposed architecture is fundamentally different, since it is inspired by GCNs. The computational complexity of NIMF is calculated as $O(|U||I|K)$. NIMF is summarized in Alg. 1. The interactions associated to test or validation ratings, are excluded from the interaction matrix $\mathbf{A}$. Afterwards, $\mathbf{A}$ is given as input to Alg. 1.

Algorithm 1 Proposed NIMF method.

Inputs: Interaction matrix $\mathbf{A}$ (consisting of the training interactions), regularization parameter λ, and number of epochs ep.

Output: Estimated rating matrix $\hat{\mathbf{R}}$.

1 Compute normalized interaction matrix, $\tilde{\mathbf{A}}$, using Eq. (6).
2 **for** $i = 1, 2, \ldots, ep$ **do**
  Compute $\hat{\mathbf{R}}$ using Eq. (9).
  Compute loss, $\mathcal{L}_{\text{NIMF}}$, using Eq. (10).
  Back propagate $\mathcal{L}_{\text{NIMF}}$.
  Update weights $\mathbf{W}_u$, $\mathbf{W}_i$.
 end for
3 Compute the final $\hat{\mathbf{R}}$ using Eq. (9).

IV. EXPERIMENTS

NIMF is evaluated on 3 real-world benchmark datasets: MovieLens 100k, MovieLens 1m, MovieLens 10m. The MovieLens datasets consist of users' ratings for movies. The prediction error is measured by the RMSE between predicted rating values and (actual) real ratings values. The activation function σ is chosen to be the non-linear RELU function:

$$\sigma(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{11}$$

that demonstrates better performance than other activation functions.

All experiments are conducted using a GPU. More specifically, the machine where the experiments were conducted consists of a 64-bit operating system with an Intel Core i9-7900X CPU at 3.3 GHz, Nvidia RTX-2080ti GPU, and 128 GB RAM.

A. Dataset Description

MovieLens 100k dataset contains $100,000$ ratings for $1,682$ movies by 943 anonymous users who rated at least 20 movies. Rating values are 1 to 5 stars with 1-star increments. MovieLens 1m dataset consists of $1,000,209$ ratings for $3,706$ movies by $6,040$ anonymous users. Each user has rated at least 20 movies. The rating scale is identical to MovieLens 100k. MovieLens 10m dataset consists of $10,000,054$ ratings for $10,681$ movies by $69,878$ anonymous users. All users are chosen to have at least 20 movie ratings each. MovieLens 10m rating values are 1 to 5 stars with 0.5-star increments (i.e., the possible rating values are 10).

The efficiency of NIMF is critically affected by the number of latent factors. Using few latent factors can make the model inefficient, as it can not learn complex functions. On the other hand, using many latent factors can lead to overfitting. The breaking points of latent factors for MovieLens datasets are depicted in Fig. 2. A similar trend in the variation of RMSE with respect to K is observed in both the validation and test set.

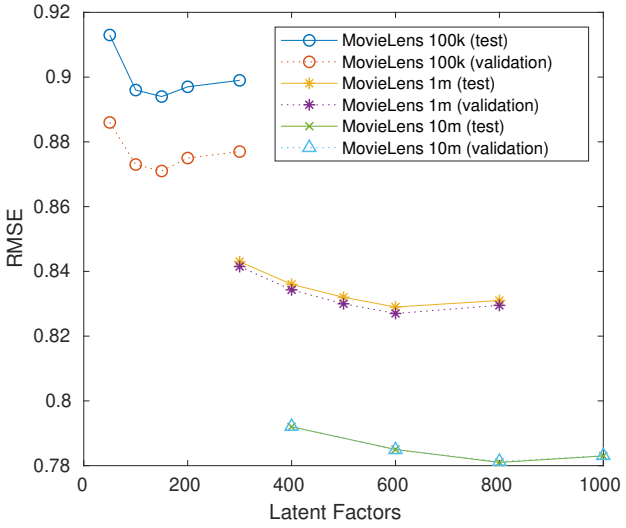

Fig. 2. Test and validation RMSE for different numbers of latent factors K on three MovieLens datasets.

B. Experiments on MovieLens datasets

MovieLens 100k dataset is split into the official training and test sets with a proportion 80%/20% of the ratings [16] in order to make a fair comparison with other methods. Afterwards, 5% of training set is used as a validation set. Therefore, training, test, and validation sets contain $76,000$, $20,000$, and $4,000$ ratings, respectively. Adaptive moment estimation (Adam)

TABLE I
TEST RMSE FOR SEVERAL METHODS ON MOVIELENS 100K DATASET. THE DATASET IS SPLIT INTO TRAINING AND TEST SETS WITH PROPORTION 80%/20% OF THE RATINGS.

Method	MovieLens 100k
MC [18]	0.973
IMC [19], [20]	1.653
GMC [21]	0.996
GRALS [22]	0.945
sRMGCNN [15]	0.929
GC-MC [13]	0.905
NIMF (ours)	**0.894**

optimizer [17] with a learning rate of 0.01 is employed. Several numbers of latent factors are tested as shown at Fig. 2. The smallest RMSE is obtained for $K = 150$. For this value of K, model training parameters are approximately 143,000. The regularization term in (10) gives preference to weight matrices with smaller squared Frobenius norm to avoid overfitting. The elements of the weight matrices are equal to $(|U| + |I|)\,K$. Considering that, the regularization parameter is set as

$$\lambda = \frac{1}{(|U| + |I|)\,K}, \tag{12}$$

which is found to offer a balance between the 2 terms in Eq. (10), which avoids both overfitting and underfitting for datasets with different size and a variety of K values. NIMF is compared against the following methods tested on MovieLens 100k dataset: Exact Matrix Completion (MC) [18], Inductive Matrix Completion (IMC) [19], [20], Matrix Completion on Graphs (GMC) [21], Graph Regularized Alternating Least Squares (GRALS) [22], sRMGCNN [15], and GC-MC [13]. The test RMSE using NIMF is 0.894. NIMF outperforms other methods, as can be seen in Table I. Although RMSE differences between GC-MC and NIMF are not statistically significant, NIMF is less computationally demanding. The training time of NIMF is 0.006 sec per epoch and test time is 0.002 sec, employing GPU. GC-MC official code can not be executed by employing the recent versions of software (i.e., CUDA 10) and hardware (i.e., Nvidia RTX-2080ti). Thus, experiments are conducted using CPU for both NIMF and GC-MC methods to make a fair comparison w.r.t. execution times. NIMF requires clearly less time than GC-MC to be executed, as shown in Table II.

TABLE II
CPU TIME REQUIREMENTS COMPARISON BETWEEN GC-MC AND NIMF METHODS.

Method	time per epoch (sec)					
	MovieLens-100k		MovieLens-1m		MovieLens-10m	
	train	test	train	test	train	test
GC-MC [13]	0.12	0.06	2.8	0.55	2137.5	5.24
NIMF (ours)	0.04	0.01	0.45	0.18	10.9	3.92

The training set of MovieLens 1m dataset comprises 90% of the ratings and the test set is created by the remaining 10% of the ratings. Model parameters are optimized over 5% of training set, which is used as validation set. Therefore, training, test, and validation sets contain $855,178$, $100,021$, and $45,010$ ratings, respectively. Adam optimizer's learning rate is set equal to 0.01. The optimal number of latent factors is chosen to be $K = 600$, as can be seen in Fig. 2. Therefore, model training parameters amount to 5.85 million. The regularization parameter is set as in Eq. (12).

TABLE III
TEST RMSE FOR SEVERAL METHODS ON MOVIELENS 1M AND MOVIELENS 10M DATASETS. DATASETS ARE SPLIT INTO TRAINING AND TEST SETS WITH PROPORTION 90%/10% OF THE RATINGS.

Method	MovieLens 1m	MovieLens 10m
PMF [23]	0.883	-
RBM [11]	0.854	0.825
BiasMF [24]	0.845	0.803
NNMF [25]	0.843	-
LLORMA [26]	0.833	0.782
I-AutoRec [27]	0.831	0.782
GC-MC [13]	0.832	0.777
CF-NADE [9]	**0.829**	**0.771**
NIMF (ours)	**0.829**	0.781

NIMF is compared against Probabilistic Matrix Completion (PMF) [23], RBM [11], Bias involved Matrix Factorization (BiasMF) [24], Neural Network Matrix Factorization (NNMF) [25], Local Low-Rank Matrix Approximation (LLORMA) [26], Item-based AutoRec (I-AutoRec) [27], GC-MC [13], and CF-NADE [9]. NIMF prediction error (i.e., RMSE=0.829) equals that of CF-NADE [9], as can be seen in Table III. NIMF possesses much less training parameters compared to the 30.48 million training parameters of CF-NADE. As a result, NIMF exhibits reduced computational complexity (i.e., 0.065 sec training time per epoch and test time equal to 0.024 sec), while CF-NADE needs 3.11 sec training time per epoch and test time equal to 0.68 sec, as shown in Table IV. NIMF attains a state-of-the-art prediction error, while its computational demands are drastically reduced.

TABLE IV
GPU TIME REQUIREMENTS COMPARISON BETWEEN CF-NADE AND NIMF METHODS.

Method	time per epoch (sec)			
	MovieLens-1m		MovieLens-10m	
	train	test	train	test
CF-NADE [9]	3.11	0.68	135.62	32.73
NIMF (ours)	0.065	0.024	0.94	0.4

MovieLens 10m dataset is split similarly to MovieLens 1m (i.e., train and test sets with proportion 90%/10% of the ratings with 5% of the training set used as validation set). Therefore, training, test, and validation sets consist of $8,550,045$, $1,000,006$, and $450,003$ ratings, respectively. Adam optimizer's learning rate is set at 0.01. Several numbers of latent factors are tested, as shown in Fig. 2. The optimal number of latent factors is $K = 800$ and model training parameters amount to 64.4 million. The regularization parameter is set, using Eq. (12). NIMF is compared against the same methods, which were tested on MovieLens 1m dataset. Although, NIMF RMSE is greater than that of CF-NADE by

0.01, the training time of NIMF is 0.94 sec per epoch and test time is 0.40 sec, while the corresponding times of CF-NADE are 135.62 sec per epoch and 32.73 sec, respectively. The time requirements per epoch between NIMF and CF-NADE are listed in Table IV.

The low time requirements of the proposed method are due to the extremely simple NIMF architecture, requiring a forward pass of 2 sparse matrix multiplications and 1 inner product. In addition, just 1 weight matrix is learned by NIMF for the entire rating scale in contrast to the learnable weight matrices of the state-of-the-art methods, such as CF-NADE and GC-MC, which are as many as the cardinality of grades in the rating scale (i.e., 5 for a 5-star rating system with 1-star increments and 10 for a 5-star rating system with 0.5-star increments).

V. CONCLUSION AND FUTURE WORK

A novel MF method coined as NIMF has been proposed. NIMF uses interactions between users and items in order to learn their latent representations. The architecture of NIMF consists of 2 layers only. One layer is responsible for users' representations and the other one concerns with items' representations. This simple architecture leads to a model with competitive performance compared to that of state-of-the-art methods. Furthermore, NIMF achieves a large train/test time reduction compared to other methods. The low time requirements of NIMF motivate future research, because it can lead to an efficient handling of new users in real-world applications, especially without retraining the model. A niche application is hotel/restaurant recommendation mining ratings and user preferences in platforms, such as booking, tripadvisor.

REFERENCES

[1] D. Agarwal and B.-C. Chen, "fLDA: matrix factorization through latent Dirichlet allocation," in *Proc. 3rd ACM Int. Conf. Web Search and Data Mining*, 2010, pp. 91–100.

[2] H. Shan and A. Banerjee, "Generalized probabilistic matrix factorizations for collaborative filtering," in *Proc. 2010 IEEE Int. Conf. Data Mining*, 2010, pp. 1025–1030.

[3] I. Fernández-Tobías, I. Cantador, P. Tomeo, V. W. Anelli, and T. Di Noia, "Addressing the user cold start with cross-domain collaborative filtering: exploiting item metadata in matrix factorization," *User Modeling and User-Adapted Interaction*, vol. 29, no. 2, pp. 443–486, 2019.

[4] S. Funk, "Netflix update: Try this at home," 2006.

[5] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2008, pp. 426–434.

[6] K. Li, X. Zhou, F. Lin, W. Zeng, and G. Alterovitz, "Deep probabilistic matrix factorization framework for online collaborative filtering," *IEEE Access*, vol. 7, pp. 56 117–56 128, 2019.

[7] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proc. 21th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2015, pp. 1235–1244.

[8] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[9] Y. Zheng, B. Tang, W. Ding, and H. Zhou, "A neural autoregressive approach to collaborative filtering," *arXiv preprint arXiv:1605.09477*, 2016.

[10] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," in *Proc. 14th Int. Conf. Artificial Intelligence and Statistics*, 2011, pp. 29–37.

[11] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. 24th Int. Conf. Machine Learning*, 2007, pp. 791–798.

[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[13] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[14] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[15] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 3697–3707.

[16] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interactive Intelligent Systems*, vol. 5, no. 4, pp. 1–19, 2015.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, p. 717, 2009.

[19] P. Jain and I. S. Dhillon, "Provable inductive matrix completion," *arXiv preprint arXiv:1306.0626*, 2013.

[20] M. Xu, R. Jin, and Z.-H. Zhou, "Speedup matrix completion with side information: Application to multi-label learning," in *Advances in Neural Information Processing Systems*, 2013, pp. 2301–2309.

[21] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Matrix completion on graphs," *arXiv preprint arXiv:1408.1717*, 2014.

[22] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Advances in Neural Information Processing Systems*, 2015, pp. 2107–2115.

[23] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems*, 2008, pp. 1257–1264.

[24] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[25] G. K. Dziugaite and D. M. Roy, "Neural network matrix factorization," *arXiv preprint arXiv:1511.06443*, 2015.

[26] J. Lee, S. Kim, G. Lebanon, and Y. Singer, "Local low-rank matrix approximation," in *Proc. Int. Conf. Machine Learning*, 2013, pp. 82–90.

[27] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 111–112.