Neural Network Coupled Matrix Factorization

Ji-Eun Lee
Seoul National University
dreamhunter@snu.ac.kr

Hyunsik Jeon
Seoul National University
jeon185@gmail.com

U Kang
Seoul National University
ukang@snu.ac.kr

***Abstract*—How can we accurately predict item ratings by users using multiple coupled data? Nowadays huge amount of sparse information is available, and in many cases auxiliary data associated with rating data are also present. Training a model to predict missing ratings is equivalent to finding a complex relationship between each user and each item. Existing methods assumed this relationship as a fixed linear function, however this causes the predictions to be biased to the mean of ratings causing information loss. Therefore it is crucial to design a flexible model that reveals hidden, non-linear relationships between users and items.**

In this paper, we propose NN-CMF, a neural network based method that predicts missing values of rating matrix by learning a non-linear function and latent matrices exploiting both rating matrix and auxiliary data. While conventional matrix factorization methods predict missing values through the inner product of latent vectors, NN-CMF learns a general non-linear function for latent vectors and therefore provides more accurate prediction. Experiments show that NN-CMF outperforms the conventional coupled matrix factorization methods by up to 5.4%. Our method is especially superior in improving accuracy on sparser datasets, making it more useful for real world applications.

***Keywords*-matrix factorization; neural networks; recommender system;**

I. Introduction

How can we accurately predict item ratings by users using multiple coupled data? In the real world, we have huge amounts of sparse data such as item rating data, social network, item characteristics data, etc. The item rating data have many of its entries missing. Predicting these missing values is equivalent to predicting the rating a user will give to an item the user has never seen before. When better items are recommended by better predictions, buyers can more easily purchase goods they prefer. Likewise, from the sellers' point of view, their revenue will increase by increased sales. By using the rating data as a starting point, utilizing other related information will help to make more accurate predictions. Therefore, effectively using the abundant amount of given information to predict item ratings has been received as an important task in the data mining community [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18].

Discovering the relationship between user and item is difficult: there are numerous hidden and unknown aspects that affect user preference. Matrix Factorization (MF) [1], [2] is a traditional method that assumes the relationship between user and item as a fixed linear function. Given a ratings matrix X, MF derives two latent matrices A and B that represent user and item features, respectively. It applies the inner product on these latent matrices that results in the predicted ratings matrix $\hat{X}$ [19], [20], [21], [22], [23]. However using a linear function induces the predictions to be skewed to the mean of ratings. Simply using a linear function cannot catch each user's preference for each item, which is not fully utilizing given information [24], [25], [26], [27]. MF also uses only sparse observed ratings matrix which makes it difficult to predict item ratings of users accurately [28]. Coupled Matrix Factorization (CMF) [3], [4] solves sparsity of the ratings data by using another auxiliary data Y to subsidize the rating prediction, as explained in detail in Section II. This auxiliary data can be social networks, such as whether a user trusts another user, or movie-genre information, etc. Given X and Y, Coupled Matrix Factorization (CMF) learns three latent matrices A, B, and G that represent user, item and some other (trustee, genre) features, respectively. As MF, CMF also models the inner product of A and B as $\hat{X}$ and inner product of B and G as $\hat{Y}$, which entails the problem of using a linear function causing information loss. Also, the predicted values of CMF tend to be clustered around the mean value as shown in Table V. Therefore it is crucial to design a flexible model that reveals hidden non-linear relationships between users and items, and decreases error by avoiding clustered output values around the mean.

In this paper, we propose NN-CMF, a neural network based method that predicts missing values of rating matrix by learning a non-linear function and latent matrices exploiting both rating matrix and auxiliary data. While conventional matrix factorization methods predict missing values through the inner product of latent matrices, NN-CMF replaces the inner product (linear function) to a multilayered feedforward neural network model (non-linear function). To build the structure of coupled neural network we combine two neural networks that each train for rating and auxiliary data with an integrated loss function. We further optimize our method to prevent predictions from converging to the mean, and to control the impact of auxiliary data to balance its importance compared to the rating data. NN-CMF predicts missing values of the ratings matrix more accurately through non-

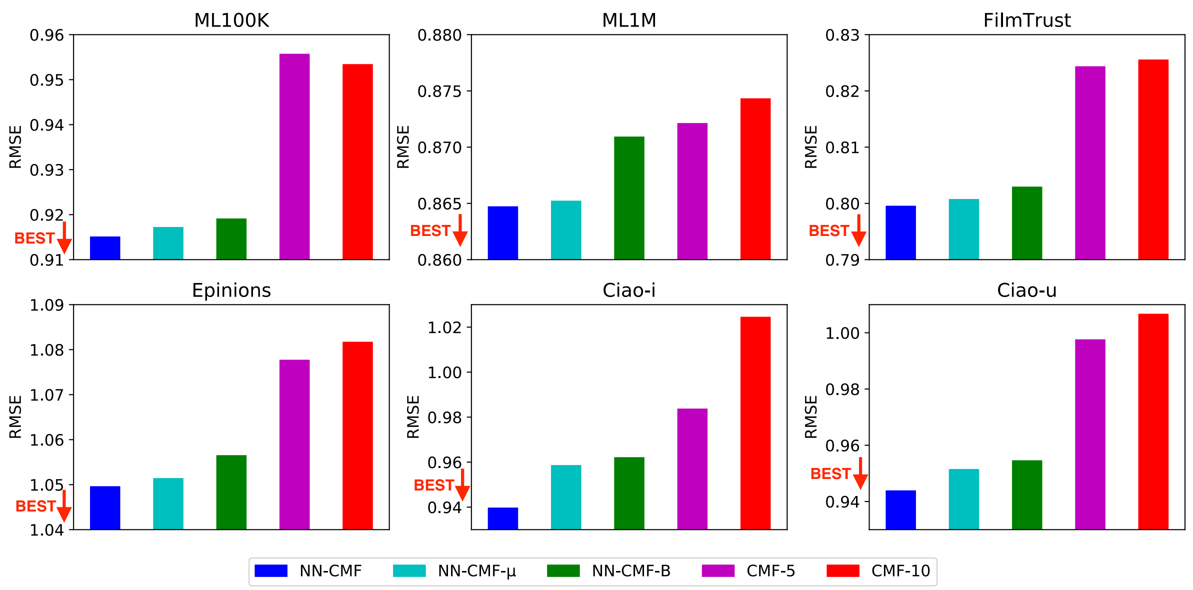

Figure 1: NN-CMF outperforms its naive versions (NN-CMF-μ and NN-CMF-B) and standard CMF, providing the minimum error.

linear mappings from factors. As shown in Figure 1, NN-CMF shows the best performance outperforming CMF and naive versions of NN-CMF.

The contributions of NN-CMF are as follows:

- **Algorithm.** We propose NN-CMF, a method to predict missing values in coupled data using neural network. Our method learns a non-linear function and latent matrices, and provides more accurate prediction.
- **Performance.** NN-CMF surpasses the accuracy of conventional coupled matrix factorization up to 5.4% and on average 3.4%. NN-CMF's predicted value distribution is closer to the real rating data distribution. Furthermore, NN-CMF show linear scalability with regard to number of ratings.
- **Overcoming Data Sparsity.** NN-CMF works even better in improving the performance on sparser datasets. NN-CMF improves the performance of CMF on sparser datasets on average by 4.2% while on denser datasets by 2.6%. In addition, as we randomly remove data points in an arbitrary dataset and thereby increase sparsity, RMSE of NN-CMF increases linearly while that of CMF increases exponentially, making the performance gap even larger.

The code of our method and datasets used in the paper are available at http://datalab.snu.ac.kr/nncmf. The rest of the paper is organized as follows. We explain preliminaries in Section II. In Section III we describe NN-CMF; we present experiment results in Section IV. After describing related works in Section V, we conclude in Section VI. Table I lists the symbols used in this paper.

II. Preliminary: Coupled Matrix Factorization

A traditional collaborative filtering method Matrix Factorization (MF) decomposes a single matrix into two latent vectors and minimizes the loss function which is based on the inner product of the two latent matrices. Since MF suffers from data sparsity problem [29], Coupled Matrix Factorization (CMF) which uses additional data has been proposed. Coupled Matrix Factorization (CMF) uses auxiliary information in addition to user-item matrix and shows better performance than MF especially when the user-item matrix is sparse.

Definition 1 (Coupled Matrix Factorization)

Assuming the auxiliary dataset used is item coupled, given a rating matrix $X \in R^{|U|\times|I|}$ and an auxiliary matrix $Y \in R^{|I|\times|C|}$, the rank d decomposition of these two primitive matrices yields three latent matrices, $A \in R^{d\times|U|}, B \in R^{d\times|I|}$ and $G \in R^{d\times|C|}$. $U, I,$ and C indicate set of users, items, and auxiliary entity, respectively. B is shared for the predicted matrices $\hat{X}$ and $\hat{Y}$ as in Equation 1:

$$\hat{X_{ui}} = A_u^\intercal B_i, \quad \hat{Y_{ic}} = B_i^\intercal G_c \tag{1}$$

where A_u, B_i, and G_c are uth column of A, ith column of B, and cth column of G, respectively. $A, B,$ and G are learned to minimize an integrated loss function. The

Table I: Table of symbols.

Symbol	Definition
U, I, C	set of users, items, and auxiliary entity (genre, trustee)
X	$X \in R^{\|U\| \times \|I\|}$ item rating matrix
X_{ui}	rating user u gave to item i
Y	$Y \in R^{\|I\| \times \|C\|}$ item coupled auxiliary matrix or $Y \in R^{\|U\| \times \|C\|}$ user coupled auxiliary matrix
Y_{ic}	information of item i for auxiliary information c (when Y is *item* coupled)
Y_{uc}	information of user u for auxiliary information c (when Y is *user* coupled)
Ω_X	set of indices of observable entries in X
Ω_Y	set of indices of observable entries in Y
f_x, f_y	feedforward network for X and Y each
A, B, G	feature matrix of user, item, and auxiliary information
A_u, B_i, G_c	feature vector for each user, item, and auxiliary information
A', B', G'	bias matrix of user, item, and auxiliary information
A'_u, B'_i, G'_c	bias vector for each user, item, and auxiliary information
d	dimension of feature vectors
d'	dimension of bias vectors
l	learning rate
λ	regularization parameter
α	parameter controlling the influence of auxiliary information
μ	average of item ratings of training set X_{train} of X
$\circ$	elementwise multiplication
L	loss function
$\sigma(x)$	sigmoid function 1/(1 + exp(-x))
$\|\ \|$	cardinality

objective function of CMF is as follows:

$$L = \frac{1}{2} \sum_{(u,i) \in \Omega_X} (\hat{X_{ui}} - X_{ui})^2 + \frac{1}{2} \sum_{(i,c) \in \Omega_Y} (\hat{Y_{ic}} - Y_{ic})^2 + \frac{\lambda}{2}(\|A\|_F^2 + \|B\|_F^2 + \|G\|_F^2), \quad (2)$$

where Ω_X and Ω_Y are sets of indices of observable entries in X and Y, respectively. λ is a regularization term to prevent overfitting. The prediction error e is defined as follows:

$$e_{ui} = \hat{X_{ui}} - X_{ui} = A_u^{\intercal} B_i - X_{ui}$$

$$e_{ic} = \hat{Y_{ic}} - Y_{ic} = B_i^{\intercal} G_c - Y_{ic}$$

The update procedures for latent vectors are as follows:

$$A_u \leftarrow A_u - l\frac{\partial L}{\partial A_u}, \qquad \frac{\partial L}{\partial A_u} = \sum_{(u,i) \in \Omega_X} (e_{ui} B_i) + \lambda A_u$$

$$B_i \leftarrow B_i - l\frac{\partial L}{\partial B_i},$$
$$\frac{\partial L}{\partial B_i} = \sum_{(u,i) \in \Omega_X} (e_{ui} A_u) + \sum_{(i,c) \in \Omega_Y} (e_{ic} G_c) + \lambda B_i$$

$$G_c \leftarrow G_c - l\frac{\partial L}{\partial G_c}, \qquad \frac{\partial L}{\partial G_c} = \sum_{(i,c) \in \Omega_Y} (e_{ic} B_i) + \lambda G_c$$

where l is a learning rate.

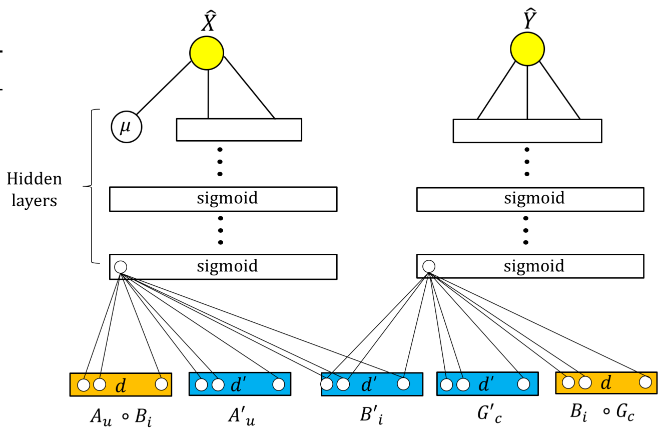

Figure 2: Neural network structure of NN-CMF for item coupled dataset. Elementwise multiplication of feature vectors (A_u, B_i, and G_c) are concatenated with bias vectors (A'_u, B'_i, and G'_c) as input. Each hidden layer is activated by sigmoid function while the last layer is linear. Detailed explanation is in Section III-A.

The coupled data type can be either user or item. The CIMF [4] uses movie-genre matrix as auxiliary information, and is more accurate than the baseline PMF [19] which uses only user-movie matrix. SoRec [3] uses social network as auxiliary information and improves the performance by 9.98% on average in terms of accuracy over the baseline PMF. In particular, it shows better performance for users with fewer item ratings, which means that CMF can get better results for sparse data. Therefore, it is crucial to utilize additional information because data is getting bigger and sparser nowadays.

III. Proposed Method

Using neural network in coupled matrix factorization allows us to learn a general non-linear function for the factorized matrices, unlike the conventional method which was constrained to inner product.

We create two networks that train for rating and auxiliary data each. NN-CMF combines the networks by a unified learning process, using an integrated loss function. We explain our base method in Section III-A. We optimize our method by learning only residuals from the mean value of the rating data. This prevents predictions from converging to the mean, which we explain in detail in Section III-B. We further optimize our method by multiplying a constant to the auxiliary information loss ($loss_Y$ of Equation 6) to reflect the importance of the additional information compared to the ratings information, which we elaborate in Section III-C.

A. *NN-CMF-B*

NN-CMF-B is the starting point of our method. We first explain the structure of our model and how our model is formulated. Then we describe the motivation regarding how we build the structure of a coupled neural network.

We will assume that the auxiliary matrix Y is an item coupled information matrix as shown in Figure 2. Given a rating matrix $X \in R^{|U| \times |I|}$ and an auxiliary matrix $Y \in R^{|I| \times |C|}$ where the item information is coupled, these two matrices are factorized into three latent feature matrices $A \in R^{d \times |U|}, B \in R^{d \times |I|}$, $G \in R^{d \times |C|}$ and three latent bias matrices $A' \in R^{d' \times |U|}, B' \in R^{d' \times |I|}$, $G' \in R^{d' \times |C|}$. U, I, and C indicate set of users, items, and auxiliary entity respectively, and each u of U, i of I, and c of C has a feature vector and a bias vector. We use feature and bias vectors to create an input vector that effectively describes the given data. The user coupled case is similarly formulated. With feature and bias vectors A_u, B_i, $G_c \in R^d$, A'_u, B'_i, $G'_c \in R^{d'}$, the model estimates each element of X and Y as follows:

$$\hat{X}_{ui} = f_x((A_u \circ B_i), A'_u, B'_i) \tag{3}$$

$$\hat{Y}_{ic} = f_y((B_i \circ G_c), B'_i, G'_c) \tag{4}$$

where f is a feedforward network, and x and y are weights used in the network. Feature vectors are multiplied to each other and bias vectors are concatenated to the multiplied feature vectors to create the input vector $\in R^{d+2d'}$. Feature vectors are multiplied to each other to constitute the interaction between entities. Bias vectors are concatenated to the multiplied feature vectors to maximize generality and allow the model to learn a non-linear function. Input vector goes through each neural network which results in a single predicted value. To learn the feature vector, bias vector, and network weights, we minimize the objective function defined in Equation 8:

$$loss_X = \frac{1}{2} \sum_{(u,i) \in \Omega_X} (\hat{X}_{ui} - X_{ui})^2 \tag{5}$$

$$loss_Y = \frac{1}{2} \sum_{(i,c) \in \Omega_Y} (\hat{Y}_{ic} - Y_{ic})^2 \tag{6}$$

$$loss_{reg} = \frac{1}{2}(\|A_u\|_2^2 + \|A'_u\|_2^2 + \|B_i\|_2^2 + \|B'_i\|_2^2 + \|G_c\|_2^2 + \|G'_c\|_2^2) \tag{7}$$

$$\begin{aligned} L &= loss_X + loss_Y + \lambda \cdot loss_{reg} = \\ &\frac{1}{2} \sum_{(u,i) \in \Omega_X} (\hat{X}_{ui} - X_{ui})^2 + \frac{1}{2} \sum_{(i,c) \in \Omega_Y} (\hat{Y}_{ic} - Y_{ic})^2 + \\ &\frac{\lambda}{2}(\|A_u\|_2^2 + \|A'_u\|_2^2 + \|B_i\|_2^2 + \|B'_i\|_2^2 + \|G_c\|_2^2 + \|G'_c\|_2^2) \end{aligned} \tag{8}$$

$loss_X$ minimizes the difference between the original existing item rating values and the predicted values. Likewise, $loss_Y$ also minimizes the difference between the original values and the predicted values regarding the auxiliary matrix. Ω_X and Ω_Y are sets of indices of observable entries in X and Y, respectively. λ is a regularization parameter multiplied to $loss_{reg}$ to prevent overfitting of feature and bias vectors. Preventing overfitting is crucial because if features and biases are not constrained, the learned model will fit too much for the training data. This makes the performance on unseen data such as the test data deteriorate.

To build the structure of coupled neural network we need to train for two different matrices, rating and auxiliary matrices, but combine the learning process so we can simultaneously update the shared matrices, B and B'. By using separate weights for $\hat{X}$ and $\hat{Y}$ we create two networks, f_x and f_y, that train for rating and auxiliary data respectively. We combine f_x and f_y by using an integrated loss function. The loss function contains both $loss_X$ and $loss_Y$ and regularizes for all components of $\hat{X}$ and $\hat{Y}$. By performing gradient descent on this loss function for feature and bias vectors we unite the learning process, and thereby fully establish coupled neural network. We explain the learning process in detail in Section III-D.

B. NN-CMF-μ

Most existing methods overlook the fact that predicted values tend to converge to the mean value. This can be prevented by learning only residuals from the mean value. In NN-CMF-μ we introduce the term μ to represent the mean of X_{train} (training set of X). We train the model with μ subtracted from the data. μ is added to the predicted rating at the last layer of the overall structure as shown in Figure 2. This means that $loss_X$ in Equation 5 equals to Equation 10, but the subtractions are canceled out so it does not show on the final objective function.

$$\mu = \frac{\sum_{(u,i) \in \Omega_{X_{train}}} X_{ui}}{|X_{train}|} \tag{9}$$

$$loss_X = \frac{1}{2} \sum_{(u,i) \in \Omega_X} ((\hat{X}_{ui} - \mu) - (X_{ui} - \mu))^2 \tag{10}$$

Doing this prevents the model always predicting the missing value as the average of observed values since the model will focus on learning how differently each user or item behaves aside the overall average.

C. NN-CMF

Conventional CMF does not notice the difference of importance between the item rating matrix and auxiliary matrix. In NN-CMF we introduce the term α to reflect the importance of the auxiliary matrix relative to the item rating matrix. α is multiplied to $loss_Y$ defined in Equation 6. Therefore the final objective function to minimize is as

Table II: Statistics of the Datasets.

dataset	# of users	# of items	# of ratings	density of ratings	# of auxiliary rows	# of auxiliary columns	#of auxiliary data	density of auxiliary matrix	total density
[a]**ML100K**	943	1682	100000	6.30%	1682	19	31958	100%	8.15%
[b]**ML1M**	6040	3706	1000209	4.47%	3883	19	73777	100%	4.77%
[c]**FilmTrust**	1508	2071	35497	1.14%	609	732	1853	0.42%	1.04%
[d]**Epinions**	40163	139738	664824	0.01%	33960	49288	487183	0.03%	0.02%
[e]**Ciao-i**	17615	16121	72665	0.03%	16121	17	274057	100%	0.12%
Ciao-u	17615	16121	72665	0.03%	1438	4299	40133	0.65%	0.04%

[a]https://grouplens.org/datasets/movielens/100k/
[b]https://grouplens.org/datasets/movielens/1m/
[c]https://www.librec.net/datasets/filmtrust.zip
[d]http://www.trustlet.org/downloaded_epinions.html
[e]https://www.librec.net/datasets/CiaoDVD.zip

follows:

$$L = loss_X + \alpha \cdot loss_Y + \lambda \cdot loss_{reg} = \frac{1}{2}\sum_{(u,i)\in\Omega_X}(\hat{X}_{ui} - X_{ui})^2 + \frac{\alpha}{2}\sum_{(i,c)\in\Omega_Y}(\hat{Y}_{ic} - Y_{ic})^2 + \frac{\lambda}{2}(\|A_u\|_2^2 + \|A'_u\|_2^2 + \|B_i\|_2^2 + \|B'_i\|_2^2 + \|G_c\|_2^2 + \|G'_c\|_2^2) \tag{11}$$

α controls the importance of the auxiliary information relative to the main rating information. This allows us to control the magnitude of influence of the auxiliary information. The auxiliary information is less influential than the item rating information when α is small and more influential when α large. Intuitively the rating information may seem to be more important for the final prediction. When the auxiliary matrix is item coupled, auxiliary information will provide how much some other item is similar or different from another item. This provides insight about whether the user will prefer an unknown item. If it is similar to items he or she used to like, it is more likely that he or she will prefer this unknown item as well. However there can be exceptions and auxiliary information cannot give more direct insight than the actual rating information itself. Same goes for when the auxiliary information is user coupled. Information about how one user is similar to another user is a secondary information compared to the rating information. However, if the ratings information is very sparse, increasing the importance of auxiliary information improves the performance. There is not enough direct information so we have to increase the magnitude of indirect information to get better predictions. In addition, if the densities of rating and auxiliary data differ a lot, we need to balance the magnitude of both data. We show introducing the α term produces better results in Section IV-B.

D. Learning Process

In total, we have two different targets to update: features, biases (A, B, G, A', B', G') and networks (f_x, f_y). Prediction error e is defined as follows :

$$e_{ui} = \hat{X}_{ui} - X_{ui} = f_x((A_u \circ B_i), A'_u, B'_i) - X_{ui}$$
$$e_{ic} = \hat{Y}_{ic} - Y_{ic} = f_y((B_i \circ G_c), B'_i, G'_c) - Y_{ic}$$

We update the features using gradient descent similar to the method stated in Gatys et al. [30] which also learns multiple features sets. We alternate between (1) updating the network while fixing features and biases, and (2) updating features and biases while fixing the network. The update procedures for (1) are conducted by back-propagation using gradient descent. The update procedures for (2) are as follows where l is a learning rate.

$$A_u \leftarrow A_u - l\frac{\partial L}{\partial A_u}, \quad \frac{\partial L}{\partial A_u} = \sum_{(u,i)\in\Omega_X}(e_{ui}\frac{\partial \hat{X}_{ui}}{\partial A_u}) + \lambda A_u$$
$$A'_u \leftarrow A'_u - l\frac{\partial L}{\partial A'_u}, \quad \frac{\partial L}{\partial A'_u} = \sum_{(u,i)\in\Omega_X}(e_{ui}\frac{\partial \hat{X}_{ui}}{\partial A'_u}) + \lambda A'_u \tag{12}$$

$$B_i \leftarrow B_i - l\frac{\partial L}{\partial B_i},$$
$$\frac{\partial L}{\partial B_i} = \sum_{(u,i)\in\Omega_X}(e_{ui}\frac{\partial \hat{X}_{ui}}{\partial B_i}) + \alpha\sum_{(i,c)\in\Omega_Y}(e_{ic}\frac{\partial \hat{Y}_{ic}}{\partial B_i}) + \lambda B_i$$

$$B'_i \leftarrow B'_i - l\frac{\partial L}{\partial B'_i},$$
$$\frac{\partial L}{\partial B'_i} = \sum_{(u,i)\in\Omega_X}(e_{ui}\frac{\partial \hat{X}_{ui}}{\partial B'_i}) + \alpha\sum_{(i,c)\in\Omega_Y}(e_{ic}\frac{\partial \hat{Y}_{ic}}{\partial B'_i}) + \lambda B'_i \tag{13}$$

$$G_c \leftarrow G_c - l\frac{\partial L}{\partial G_c}, \quad \frac{\partial L}{\partial G_c} = \alpha \sum_{(i,c)\in\Omega_Y} (e_{ic}\frac{\partial \hat{Y}_{ic}}{\partial G_c}) + \lambda G_c$$
$$G'_c \leftarrow G'_c - l\frac{\partial L}{\partial G'_c}, \quad \frac{\partial L}{\partial G'_c} = \alpha \sum_{(i,c)\in\Omega_Y} (e_{ic}\frac{\partial \hat{Y}_{ic}}{\partial G'_c}) + \lambda G'_c \tag{14}$$

IV. Experiments

We present experiment results of NN-CMF evaluating the following questions:

- **Q1. (Performance of NN-CMF)** How accurate is NN-CMF? (Section IV-B)
- **Q2. (Comparison with Baselines)** How better is NN-CMF compared to the baselines CMF-5 and CMF-10? (Section IV-C)
- **Q3. (Scalability)** How scalable is NN-CMF? (Section IV-D)
- **Q4. (Performance by Data Sparsity)** How does NN-CMF perform regarding data sparsity? (Section IV-E)

We show overall results compared with naive versions of NN-CMF in Section IV-B and baselines in Section IV-C. We state the scalability of our method in Section IV-D, and performance by data sparsity in Section IV-E.

A. Experimental Setup

Data. We conduct experiments on real-world datasets MovieLens 100K (ML100K), MovieLens 1M (ML1M), FilmTrust, Epinions, and Ciao. The statistic of the datasets is summarized in Table II. ML100K and ML1M datasets provide item-coupled information (movie-genre), while FilmTrust and Epinions datasets provide user coupled information (user-user trust relationship). Ciao provides both item-coupled and user-coupled information which we denote by Ciao-i and Ciao-u respectively in the results. Density of ratings is computed as follows:

$$\frac{|ratings|}{|U|\cdot|I|} \tag{15}$$

In item coupled datasets, ML100K has the highest density, ML1M the second highest, and Ciao-i has the lowest density. In user coupled datasets, FilmTrust has the highest density, Ciao-u the second highest density, and Epinions has the lowest density.

Metrics. We use root mean square error (RMSE) as the metric for our experiments.

$$\text{RMSE} = \sqrt{\frac{\sum_u \sum_i (\hat{X}_{ui} - X_{ui})^2}{|ratings|}}$$

A smaller RMSE indicates better performance since it means the predicted value is closer to the real value. We additionally state the variance of prediction value distribution compared to the original distribution of rating values. Comparing the distribution of predictions shows the accuracy of a method in more detail, while RMSE averages all errors so such details are overlooked.

Table III: Common parameter setting of NN-CMF.

hidden layers	learning rate (l)	dropout rate	d	d'
$[10]*4$	0.0015	0.0	15	5

Table IV: Parameter setting of NN-CMF.

	batch size	lambda (λ)	alpha (α)
ML100K	4000	1.0	0.7
ML1M	40000	1.0	1.4
FilmTrust	2000	1.5	0.6
Epinions	20000	2.0	1.5
Ciao-i	8000	2.0	0.1
Ciao-u	10000	1.0	0.6

Baselines. We use conventional CMF using rank 5 and 10, which we denote by CMF-5 and CMF-10, respectively. We apply sigmoid function to the predicted matrices from Equation 1 as below:

$$\hat{X_{ui}} = \sigma(A_u^{\intercal} B_i), \ \ \hat{Y_{ic}} = \sigma(B_i^{\intercal} G_c) \tag{16}$$

Many existing methods that deal with sparse and large datasets with values within a narrow range uses the sigmoid function $\sigma(x) = 1/(1 + exp(-x))$ to bound the range of predictions [19] [3]. This function prevents the loss from diverging when using such datasets. Sigmoid function bounds the values to be between 0 and 1, while rating values range from 1 to 5. Without loss of generality we normalize the rating values of X using min-max normalization: $(r-1)/(5-1)$ for rating r. We show that the performance of our proposed NN-CMF surpasses the performance of both CMF-5 and CMF-10.

Parameter. We find the best parameter setting using binary search for both NN-CMF and conventional CMF. The parameters we use for each dataset are shown in Tables III and IV.

B. Performance of NN-CMF

We use 5-fold cross validation for experiments. We randomly shuffle the data set and split it into five folds. For each experiment we use each fold as test set and the other four folds as training set. We also use 2 percent of the training set as valid set for the learning process. We continue the learning process until the RMSE of the validation set has increased continuously. Then we bring the saved model when the RMSE of the validation set was the smallest and report the resulting RMSE of the test set on the final model. The main results of NN-CMF are shown in Figure 1.

NN-CMF-μ always outperforms NN-CMF-B proving the usefulness of modeling the term μ. We find that using μ prevents the predictions from converging to the mean. Table V shows the variance of prediction distribution of methods

Table V: Variance of prediction distribution of methods on ML100K dataset. Methods that use the term μ, NN-CMF and NN-CMF-μ, show the closer variance to the original rating variance 1.2671, and NN-CMF shows the closest variance to the original.

method	variance
NN-CMF	0.4987
NN-CMF-μ	0.4830
NN-CMF-B	0.4745
CMF	0.2341

Table VI: Difference of the best value of α from 1 regarding rating data density. Datasets are ordered from denser sets to sparser sets. The difference increases as rating data gets sparser.

item-coupled	difference	user-coupled	difference
ML100K	0.3	FilmTrust	0.4
ML1M	0.4	Ciao-u	0.4
Ciao-i	0.9	Epinions	0.5

on ML100K dataset. Methods that use the term μ, NN-CMF and NN-CMF-μ, show the closer variance to the original rating variance 1.2671, and NN-CMF shows the closest variance to the original. This indicates our method produces the most similar rating predictions to the original.

NN-CMF surpasses NN-CMF-μ at all times, which validates the effectiveness of using the term α. We also find that the difference of α from 1 and data sparsity have a positive relationship. Recalling that when α is 1, it is equivalent to not using the α term, the value of α tends to deviate from 1 as data gets sparser. We examine the relationship in item-coupled datasets and user-coupled datasets, separately. Item-coupled datasets have 100% density of auxiliary data, while user-coupled datasets have much more sparse coupled data. This difference influences the size of α, which made us consider the two types of datasets separately. We present the difference of α from 1 in Table VI. For each type of coupling, datasets are ordered from denser to sparser sets. For both item-coupled and user-coupled datasets, the difference increases as rating data sparsity rises. The result implies that for sparser datasets it is more crucial to balance the impact of auxiliary data. In summary, using α is effective for solving data sparsity problem.

All additional optimizations improve the performance of NN-CMF. On average additional optimizations increase the performance by 0.9% and up to 2.3%. For sparser datasets additional optimizations increase performance on average by 1.4% and for denser datasets by 0.5%, indicating μ and α are even more effective on handling *sparser* datasets.

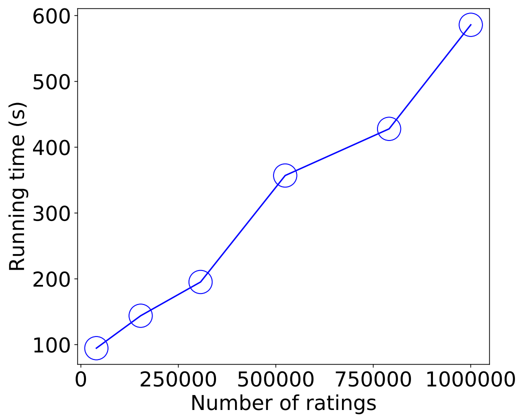

Figure 3: The running time of NN-CMF linearly increases with regard to number of ratings.

C. *Comparison with Baselines*

We compare the performance of NN-CMF with baseline methods, CMF-5 and CMF-10. Results are shown in Figure 1. NN-CMF outperforms both CMF-5 and CMF-10 for all datasets, at most by 5.4% and on average 3.4%. NN-CMF improves the performance of CMF on sparser datasets on average by 4.2% while on denser datasets by 2.6%. This shows that NN-CMF is even more effective in improving the performance of sparse datasets, making it more useful in real world applications.

NN-CMF also shows superior prediction distribution compared to CMF. As shown in Table V, The variance of NN-CMF is much closer to the original variance, than that of CMF is.

D. *Scalability*

We show the scalability of NN-CMF by reporting the running time on principal submatrices of ML1M dataset. We vary the range of the number of ratings from 40000 to 1000000. Running time of NN-CMF linearly increases with regard to the number of ratings, indicating our method is scalable considering data size.

E. *Performance by Data Sparsity*

We compare how NN-CMF and CMF perform considering data sparsity. Since we have two different sizes of matrices with different densities for each dataset, we calculate the total density considering the size of each matrix as well. The total density of an item-coupled dataset is computed as follows:

$$Total\ density = \frac{|ratings| + |auxiliary\ data|}{(|U| \cdot |I|) + (|I| \cdot |C|)} \quad (17)$$

For a user-coupled dataset the term $|I| \cdot |C|$ changes to $|U| \cdot |C|$. Total sparsity would be $1 - total\ density$. We use the most dense dataset ML100K to show our results.

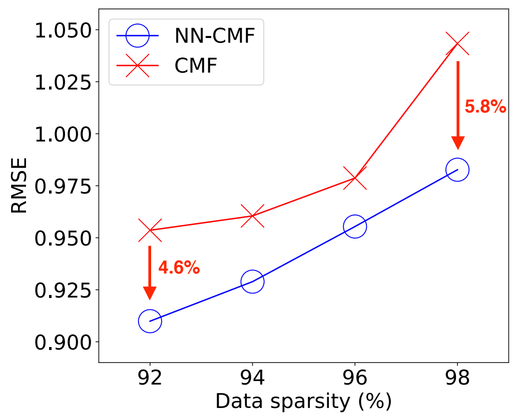

Figure 4: RMSE of NN-CMF linearly increases as sparsity rises while that of CMF increases exponentially.

The total sparsity of ML100K is approximately 92%. We randomly remove 25%, 50%, and 75% of the data, making the total sparsity 94%, 96%, and 98% respectively. The performance of NN-CMF and CMF regarding data sparsity is shown in Figure 4. As sparsity increases in the dataset, NN-CMF outperforms CMF from 4.6% to 5.8%. Also RMSE of NN-CMF linearly increases as sparsity rises while that of CMF increases exponentially. This indicates that NN-CMF utilizes given data better and maintains accuracy.

V. Related Works

We review related works based on two perspectives: collaborative filtering with additional data, and neural network matrix factorization.

A. Collaborative Filtering with Additional Data

Many studies proposed methods that leverage additional information to alleviate the rating sparsity problem in collaborative filtering. The methods are classified into two categories according to the type of additional data: user based methods, and item based methods.

User based methods. Studies [1], [3], [5], [6], [7], [9] extensively proved that using not only rating data but also user based data mitigates rating sparsity problem in collaborative filtering. Ma et al. [3] proposed SoRec that uses rating matrix and trust matrix to predict unobserved ratings. SoRec optimizes item latent vector using rating matrix, and jointly optimizes using both rating matrix and trust matrix. SoRec predicted the ratings more precisely than the baseline models using the user's in-node and out-node ratio. Yang et al. [7] proposed a method that extracts a subset of friends to be used in collaborative filtering. Guo et al. [5] proposed TrustSVD that combines user's implicit data and rating matrix based on biased matrix factorization [2].

Item based methods. Previous works [4], [31], [8], [26], [25], [32], [33] proposed models that utilize item based data as additional information, and there also have been many methods using review data to provide personalized recommendations [34]. Li et al. [4] used implicit data of items to optimize item latent vector. Leung et al. [31] proposed a method that quantifies reviews through sentiment analysis and reflects it in rating prediction. Wang et al. [8] combined topic modeling and collaborative filtering. Wang et al. [26] integrated Stacked Denoising AutoEncoder (SDAE) [35] and Probabilistic Matrix Factorization (PMF) [19]. Kim et al. [25] proposed ConvMF that incorporate Convolutional Neural Network (CNN) model which captures contextual information of the item documents into the PMF. ConvMF learned item latent vector better through the CNN model and predicted ratings more accurately than existing baseline models. Kim et al. [36] further improved the accuracy of ConvMF by differently applying the Gaussian noise of each item. Hu et al. [32] proposed a model that integrates item reviews into Matrix Factorization based Bayesian personalized ranking (BPR-MF). Bauman et al. [33] proposed SULM that analyzes sentiment for each aspect by decomposing reviews into aspect units. SULM predicts not only the probability that a user likes an item but also which aspect has a big influence.

B. Neural Network Matrix Factorization

Dziugaite and Roy [24] proposed Neural Network Matrix Factorization (NNMF) that performs matrix factorization using neural network. Conventional matrix factorization predicts entries of a matrix with the inner product of two vectors. NNMF replaces the linear function to a multilayered feed forward neural network. However NNMF is limited in the following aspects: 1) it cannot utilize additional information, and thus cannot perform coupled matrix factorization, 2) the model uses dense connections in the first layer which limits the performance when data are sparse, and 3) the model outputs a value biased toward the mean value since the model does not consider the residual from the mean.

VI. Conclusion

In this paper, we propose NN-CMF, a neural network based method that predicts missing values of rating matrix by learning a non-linear function and latent matrices from both rating matrix and auxiliary data. NN-CMF is a generalized implementation of coupled matrix factorization using neural network. NN-CMF surpasses the accuracy of conventional coupled matrix factorization by up to 5.4% and on average by 3.4%. NN-CMF's predicted value distribution is closer to the real rating data distribution. Furthermore, NN-CMF shows linear scalability on the number of ratings. NN-CMF works even better in improving the performance on sparser datasets. NN-CMF improves the performance of CMF on sparser datasets on average by 4.2% while on denser datasets by 2.6%. In addition, as we randomly remove data in an arbitrary dataset thereby increase sparsity, RMSE of NN-CMF increases linearly while CMF increases exponentially, making the performance gap even larger. Future works include modeling temporal dynamics of recommendation with recurrent neural network.

References

[1] A. J. Chaney, D. M. Blei, and T. Eliassi-Rad, "A probabilistic model for using social networks in personalized item recommendation," in *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 2015, pp. 43–50.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems."

[3] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: social recommendation using probabilistic matrix factorization," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, 2008, pp. 931–940.

[4] F. Li, G. Xu, and L. Cao, "Coupled item-based matrix factorization," in *Web Information Systems Engineering - WISE 2014 - 15th International Conference, Thessaloniki, Greece, October 12-14, 2014, Proceedings, Part I*, 2014, pp. 1–14.

[5] G. Guo, J. Zhang, and N. Yorke-Smith, "Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[6] G. Guo, J. Zhang, D. Thalmann, and N. Yorke-Smith, "Etaf: An extended trust antecedents framework for trust prediction," in *Proceedings of the 2014 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2014.

[7] X. Yang, H. Steck, and Y. Liu, "Circle-based recommendation in online social networks," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 1267–1275.

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

[9] H. Ma, "An experimental study on implicit social recommendation," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 73–82.

[10] H. Fang, Y. Bao, and J. Zhang, "Leveraging decomposed trust in probabilistic matrix factorization for effective recommendation," in *In Proceedings of the 28th AAAI Conference on Artificial Intelligence*. AAAI, 2014, pp. 30–36.

[11] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *In Proceedings of the 4th ACM Conference on Recommender Systems*. RecSys, 2010, pp. 135–142.

[12] B. Yang, Y. Lei, D. Liu, and J. Liu, "Social collaborative filtering by trust," in *In Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. IJCAI, 2013, pp. 2747–2753.

[13] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *In Proceedings of the 4th ACM International conference on Web Search and Data Mining*. WSDM, 2011, pp. 287–296.

[14] Y. Zheng, B. Mobasher, and R. Burke, "User-oriented context suggestion," in *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization (UMAP 16)*. ACM, 2016, pp. 249–258.

[15] G. Ganu, Y. Kakodkar, and A. . Marian, "Improving the quality of predictions using textual information in online user reviews." Inf. Syst. 38, 2013, pp. 1–15.

[16] N. Hariri, B. Mobasher, R. Burke, and Y. Zheng, "Context-aware recommendation based on review mining." ITWP, 2011.

[17] G. Ling, M. R. Lyu, and I. King, "Ratings meet reviews, a combined approach to recommend," in *Proceedings of the 8th ACM Conference on Recommender systems*, 2014.

[18] S. Pero and T. Horvath, "Opinion-driven matrix factorization for rating prediction," in *Springer Berlin Heidelberg*, 2013, pp. 1–13.

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

[20] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model." in *In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD, 2008, pp. 426–434.

[21] L. W. Mackey, A. S. Talwalkar, and M. I. Jordan, "Divide-and-conquer matrix factorization," in *In Advances in Neural Information Processing Systems*. NIPS, 2011.

[22] J. Lee, S. Kim, G. Lebanon, and Y. Singer., "Local low-rank matrix approximation," in *In Proceedings of the International Conference on Machine Learning*, 2013.

[23] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie., "Autorec: Autoencoders meet collaborative filtering," in *In Proceedings of the 24th International Conference on World Wide Web*. WWW, 2015.

[24] G. K. Dziugaite and D. M. Roy, "Neural network matrix factorization."

[25] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 233–240.

[26] H. Wang, N. Wang, and 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.

[27] N. D. Lawrence and R. Urtasun, "Non-linear matrix factorization with gaussian processes," in *In Proceedings of the International Conference on Machine Learning*, 2009.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[29] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, p. 4, 2009.

[30] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," 2015.

[31] C. W. Leung, S. C. Chan, and F.-l. Chung, "Integrating collaborative filtering and sentiment analysis: A rating inference approach," in *Proceedings of the ECAI 2006 workshop on recommender systems*, 2006, pp. 62–66.

[32] G.-N. Hu and X.-Y. Dai, "Integrating reviews into personalized ranking for cold start recommendation," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2017, pp. 708–720.

[33] K. Bauman, B. Liu, and A. Tuzhilin, "Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 717–725.

[34] L. Chen, G. Chen, and F. Wang, "Recommender systems based on user reviews: the state of the art," *User Modeling and User-Adapted Interaction*, vol. 25, no. 2, pp. 99–154, 2015.

[35] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." JMLR, 2010, p. 33713408.

[36] D. Kim, C. Park, J. Oh, and H. Yu, "Deep hybrid recommender systems via exploiting document context and statistics of items," *Information Sciences*, vol. 417, pp. 72–87, 2017.