mathematics

MDPI

Article

Neural Metric Factorization for Recommendation

Xiaoxin Sun [1], Liqiu Gong [1], Zhichao Han [1], Peng Zhao [1], Junchao Yu [1] and Suhua Wang [2,*]

[1] School of Information Science and Technology, Northeast Normal University, Changchun 130117, China; sunxx772@nenu.edu.cn (X.S.); gonglq659@nenu.edu.cn (L.G.); hanzc581@nenu.edu.cn (Z.H.); zhaop268@nenu.edu.cn (P.Z.); yujc248@nenu.edu.cn (J.Y.)

[2] Institute of Technology Changchun Humanities and Sciences College, Changchun 130117, China

* Correspondence: wangsuhua@ccrw.edu.cn; Tel.: +86-431-84537209

Abstract: All current recommendation algorithms, when modeling user–item interactions, basically use dot product. This dot product calculation is derived from matrix factorization. We argue that an inherent drawback of matrix factorization is that latent semantic vectors of users or items sometimes do not satisfy triangular inequalities, which may affect the performance of the recommendation. Recently, metric factorization was proposed to replace matrix factorization and has achieved some improvements in terms of recommendation accuracy. However, similar to matrix factorization, metric factorization still uses a simple, linear fashion. In this paper, we explore leveraging nonlinear deep neural networks to realize Euclidean distance interaction between users and items. We propose a generic Neural Metric Factorization Framework (NMetricF), which learns representations for users and items by incorporating Euclidean metric factorization into deep neural networks. Extensive experiments on six real-world datasets show that, compared to the previous recommendation algorithms based purely on rating data, NMetricF achieves the best performance.

Keywords: neural metric factorization; euclidean distance; collaborative filtering; deep learning

Citation: Sun, X.; Gong, L.; Han, Z.; Zhao, P.; Yu, J.; Wang, S. Neural Metric Factorization for Recommendation. *Mathematics* **2022**, *10*, 503. https://doi.org/10.3390/math10030503

Academic Editors: Pasquale De Meo and Christophe Guyeux

Received: 16 December 2021
Accepted: 28 January 2022
Published: 4 February 2022

1. Introduction

Servers and web pages connected to the Internet show an explosive trend. Massive amounts of information are presented to us at the same time. YouTube, for example, attracts millions of people around the world, with billions of hours of video played every day. There are tens of thousands of movies on Netflix, millions of books on Amazon and hundreds of millions of products on Taobao. Traditional search algorithms can only present users with the same results of sorting items but can not provide corresponding services for different users' interests and hobbies. In order to overcome the information failure caused by information overload, the personalized recommendation system came into being. For each specific user, the personalized recommendation algorithm evaluates the user's preferences based on his online interaction history (e.g., purchase, click, browse, comment, etc.), combined with some auxiliary information (e.g., attributes, social connections, text, knowledge graph, etc.). Finally, it selects the N items from the candidate set and recommends them to the user.

The recommender system essentially models the user's behavioral history, item attributes and some contextual information to infer the user's interests and recommend items. Therefore, the practical recommendation algorithm needs strong expansibility and can conveniently integrate various auxiliary information. In the process of recommendation, the most important information undoubtedly is user–item rating data among all kinds of information because rating data is an explicit feedback that most directly reflects the user's preferences. However, rating information is always sparse; therefore, rating prediction has become one of the natural objectives of the recommender system. Rating prediction is to infer unknown ratings based on the users' existing ratings of the items, so as to achieve the recommendation according to predicting ratings. Some famous business media

sites, such as Netflix, Douban, IMDb, etc., consciously collect user ratings to help with recommendations.

By factoring the rating matrix, the user latent semantic matrix and item latent semantic matrix can be obtained to represent the features of all users and all items, respectively. The product of the two matrices is used to model the user's predicted ratings for the item. Neural networks have been proved to have the ability to solve the practical problems of nonlinear ambiguous functions. They have performed well in many domains, such as image processing, speech recognition, and text categorization, etc. In addition, some researchers combined them to predict the ratings and also received good results. However, due to the defects of matrix factorization, this research still needs to be promoted. Here, we will study applying neural metric factorization in rating prediction.

The dot product does not satisfy the triangular inequality, so it means the matrix factorization can not obtain the optimal solution and reduces its performance. In view of this defect, we use a neural network for the matrix to realize rating predictions.

The main contributions of this paper :

- By reversing the rating matrix, we switch the research perspective on user–item ratings to that of user–item distance. That is, we no longer study the degree of match between user preferences and item characteristics, but instead study the distance between them.
- We propose a Neural Metric Factorization (NMetricF) model. Instead of using inner products to express user–item similarity in previous papers, Euclidean distance is used to measure user–item distance. The distance matrix predicted by this method is finally inverted again and used as the prediction score matrix.
- We have performed extensive comparative experiments and analyses for our model and the baseline models, and the results show that NMetricF outperforms the baseline approaches for both rating prediction and item sorting tasks. In particular, NMetricF's generalization ability is even better.

This paper is organized as follows: in Section 2, we describe some research work related to our study, and in Section 3, we give some preparatory knowledge. In Section 4, we show our recommendation model and explain each part in detail; in Section 5, we compare and analyze the recommendation performance with the corresponding baseline methods on two types of tasks: rating prediction and item ranking, respectively. Finally, in Section 6, we summarize our work and provide an outlook on future research.

2. Related Work

The main tasks of the recommender system include rating prediction, item ranking and next-item recommendation, which are widely used in specific applications, such as library book recommendation, academic paper citation recommendation, friend prediction or recommendation in social networks, product recommendation or rating prediction in e-commerce. In all situations, the interaction information (e.g., rating matrix) is often very sparse, and the sparsity sometimes reaches more than 90%. The rating prediction is essentially a matrix filling task, and how to predict missing values as accurately as possible through the information with less data is the key problem. Solutions to some similar problems will be introduced here.

Earlier classical work such as matrix factorization [1–3] tried to factorize a user–item implicit feedback to obtain user feature matrix and item feature matrix. Further research work also includes: Sanjay [4] proposed a probabilistic matrix factorization model that unites topic modeling and social networks. This is a hierarchical Bayesian model that automatically infers useful potential topics and social information and their importance for collaborative filtering from training data. Wang [5] explained the scores and words obtained by using implicit data, integrated collaborative filtering into probability modeling, and recommended scientific articles to online users. Lee [6] assumes that some parts of the matrix are low rank. Under the guidance of this idea, such a matrix is used to fit the scoring data in the recommender system. Kim [7] generated feature vectors of items

by convolutional neural networks with the help of auxiliary text information of items (e.g., movie plot text), and then the convolution network and MF are fused to achieve the purpose.

Although matrix factorization has had a lot of achievements, it has the drawback that the dot product cannot satisfy all inequality relations between vectors at the same time, which greatly limits its performance (we will explain this defect in Section 3.1). Therefore Dziugaite [8], He [9], and Wu [10] have proposed different models to alleviate this problem. However, these algorithms are all still based on the idea of the matrix factorization algorithm or the combination of matrix factorization and neural network, which cannot fundamentally avoid the defect of matrix factorization. Zhang [11] proposed a method of substituting metric factorization for matrix factorization, which can bypass the dot-product and satisfy triangular inequalities. Hsieh [12] proposed a collaborative metric learning algorithm; this method extends metric factorization to collaborative filtering.

3. Preliminaries

In this section, we gradually introduce the idea of neural metric factorization in the following steps. Furthermore, we clarify the context from traditional matrix factorization to neural metric factorization.

3.1. Drawback of Matrix Factorization

The matrix factorization is essentially the inner product of latent vectors. Compared with the initial matrix, the decomposed vector is very small, and the feature representation ability of the latent vector is greatly discounted. Meanwhile, the representation process of the inner product is too simple. For example, assuming that we have four users: U_1, U_2, U_3, U_4, we use $\mathrm{Sim}(m, n)$ to represent the similarity between U_m, U_n, the similarities based on interaction matrix are: $\mathrm{Sim}(2,3) > \mathrm{Sim}(1,2) > \mathrm{Sim}(1,3)$, and $\mathrm{Sim}(4,1) > \mathrm{Sim}(4,3) > \mathrm{Sim}(4,2)$. After matrix factorization, the four vectors are embedded into the low-dimensional space (here, we take two-dimension as an example for drawing). We use the vector's included angle to represent similarity; the smaller the angle, the greater the similarity, as shown in Figure 1. We find that under the condition that U_1, U_2, U_3 satisfy the first inequality, no matter how U_4 is positioned, it cannot satisfy the second inequality. Therefore, although matrix factorization achieves low-dimensional embedding and simplifies calculation, it also loses some semantic relations, which makes the feature representation contradictory.

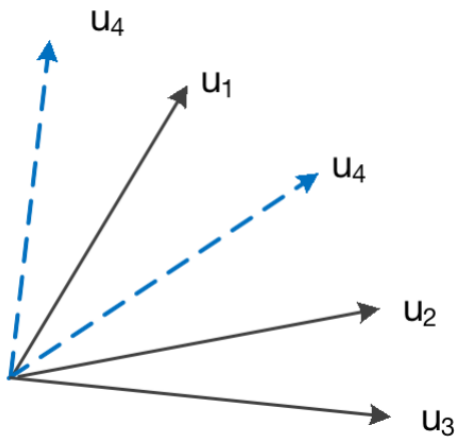


Figure 1. Defects in matrix factorization.

3.2. Metric Factorization

The idea in Reference [11] is as follows: The rating matrix of the user–item (range 1–5 of the score) is first flipped (subtract each score from the maximum score of 5), the

obtained matrix is called the distance matrix. For example, in the rating matrix, assuming that the user's rating for the item is 5, which is changed to 0 by flipping operation (5-5). That is to say, the distance between u and i is 0. If u's rating for i is 1, after flipping the operation (5-1), it becomes 4; that is to say, the distance between u and i is 4. The logical meaning of this approach is that if a user scores an item highly, it means that they have some similarity in the attributes of interest, so the distance is small; on the contrary, if a user scores an item extremely low, it means that he does not like the item, then the user and the item are not similar in the attributes of interest, so the distance is far away. Now consider the positions of U_1, U_2, U_3, U_4 in distance. Because $\mathrm{Sim}(2,3) > \mathrm{Sim}(1,2) > \mathrm{Sim}(1,3)$, U_2 is the closest to U_3 in Euclidean distance, followed by distance between U_2 and U_1, and the biggest is the distance between U_1 and U_3. We use the Euclidean distance between two vectors to represent similarity; the smaller the Euclidean distance, the greater the similarity, as shown in Figure 2. Then, because $\mathrm{Sim}(4,1) > \mathrm{Sim}(4,3) > \mathrm{Sim}(4,2)$, that is, U_4 is nearest to U_1, followed by the distance to U_3, and the farthest distance to U_2, we use a red ring (minimum radius) to express the possible positions of U_4 satisfying the minimum distance to U_1, a yellow ring (centered in radius) to indicate the possible position of U_4, which satisfies the distance from U_3, and a green ring (largest radius) to indicate the possible position of U_4, which satisfies the maximum distance from U_2. The coincidence region of the three rings is the positions of U_4, which satisfies the relationship of $\mathrm{Sim}(4,1) > \mathrm{Sim}(4,3) > \mathrm{Sim}(4,2)$. This shows that the Euclidean distance is more reasonable than the angle in describing relationships between vectors in low-dimensional space.

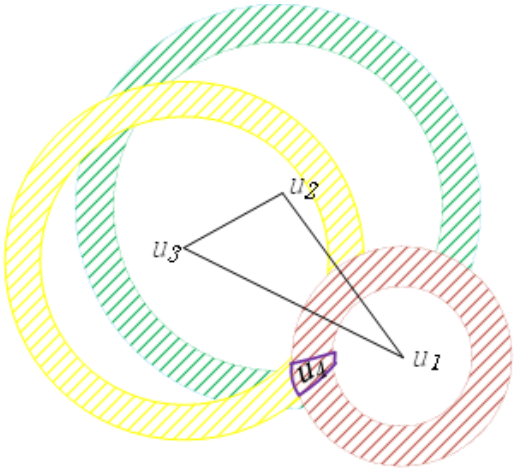


Figure 2. Using Euclidean distance to describe vectors.

Next, we explain metric factorization, but first, we flip the rating matrix into a metric (or distance) matrix:

$$D(u,i) = Max(R) - R(u,i) \tag{1}$$

In the dataset used in our experiment, the maximum user rating for items is 5, so the $Max(R)$ here is 5. The following illustrative example:

If we regard the rating matrix as a similarity matrix, that is to say, the user's rating on the item is the similarity between the user's interest and the item's attribute, then similarly, each observation value in the metric (or distance) matrix formed after flipping can be regarded as the distance between the user's interest and the item's attribute. The metric matrix is then decomposed into a low-dimensional user latent semantic matrix U and a distance operation of a low-dimensional item latent semantic matrix, that is, $D(u,i) = U - I$. For example, the user2–item3 distance in Figure 3 is:

$$d_{u,i} = D(2,3) = U_2 - I_3 = \sqrt{(U_2^1 - I_3^1)^2 + (U_2^2 - I_3^2)^2} \tag{2}$$

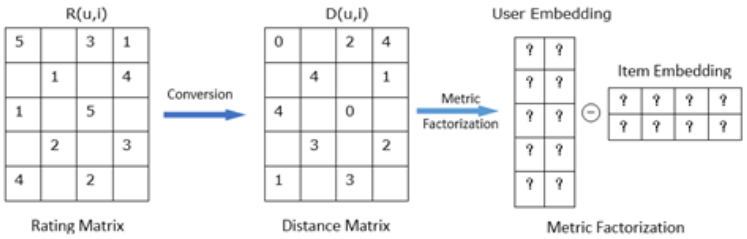


Figure 3. Rating matrix is turned into a distance matrix, and then, metric factorization is performed.

Among them, U_2 represents the second line of U, which is the latent semantic vector of user2, I_3 is the third column of I, which is the latent semantic vector of item3. $U_2 - I_3$ is the Euclidean distance between the latent semantic vectors of user2 and item3. This is the process of Metric Factorization.

3.3. Why Do We Introduce Neural Metric Factorization

We find that although Metric Factorization (FML [11]) is superior to the traditional Matrix Factorization (MF) in rating prediction, its overall effect is still not good enough. Although Euclidean distance is introduced to metric factorization instead of inner product, it is only a simple linear combination of square deviation in every dimension of the latent semantic vector, without introducing a nonlinear relationship, and it fails to consider the different levels of importance of each dimension (the weight of each dimension is the same), so it is also true. Thus, it is impossible to characterize the complex interactions. Thus, a more general framework of metric factorization is implemented by using a deep neural network. By introducing a nonlinear activation function and deeper neural layers based on the original linear computation, the expressive ability of the model can be better improved. Next, we will describe this method in detail.

4. Proposed Methodology

In this section, we introduce our neural metric factorization (NMetricF) model. Figure 4 shows the implementation model of the NMetricF method. Suppose there are M users and N items, and the ratings of user u on item i are denoted as $r \in R^{M*N}$. The rating matrix R is always sparse, i.e., many entries in R are missing, and one of the tasks of the recommendation system is to derive (or predict) the missing entries in R. To this end, the idea of our proposed method is that first, we convert the rating matrix into a distance matrix D^{M*N} by using Equation (1). Although D is still a sparse matrix, there are still some user–item distance entries. Then, the model in Figure 1 is fitted for the existing entries in D. This process is able to incidentally learn the embeddings of all users and all items, and further, the distance interactions of the learned user and item embeddings are used to estimate the missing entries in D. In this way, all the missing entries in D are filled. Finally, D is then re-inverted into an estimate of R, which is the predicted rating matrix.

Then, starting from the input layer, we introduce the model's input layer, embedding layer, metric interaction layer, nonlinear hiding layers, and output layer in detail. In the process, we have proved that Metric Factorization is one case of NMetricF in a simple form; in other words, our NMetricF model is the generalization of Metric Factorization. Finally, we provide the loss functions and optimization settings for our model on two different tasks.

4.1. General Framework

Figure 4 is the general framework of NMetricF. Generally speaking, our model combines the computation of metric factorization and the powerful learning ability of deep nonlinear neural networks to model sparse data. Therefore, it is a machine learning engineering for real-valued vectors. Let us start with the input layer and introduce the model in detail.

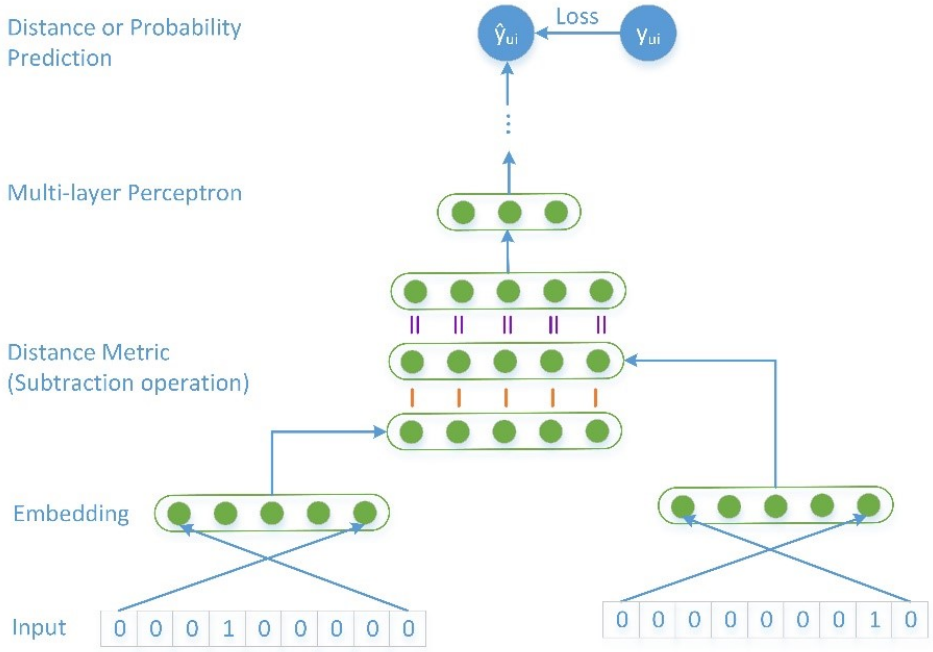


Figure 4. The framework of neural metric factorization, where $\hat{y}_{u,i}$ represents the predicted distance in the rating prediction task or the predicted probability in the item ranking task. $y_{u,i}$ represents the true distance in the rating prediction task or the true hit probability in the item ranking task.

4.2. Input Layer

Suppose the dataset is a sparse matrix, and it is rating one. As shown in Figure 4, the user's ID is converted to the m-dimensional one-hot vector U(u$\in$ R^m) as the user's input at the bottom left of Figure 4, and the item's ID is converted to the n-dimensional one-hot vector i(i$\in$ R^n) as the item's input at the bottom right of Figure 4. One-hot vector can uniquely identify each user or item but cannot contain semantic information. As a simple example, there are three one-hot values, specifically they are a = [1,0,0], b = [0,1,0], c = [0,0,1]. By calculating the Euclidean distance between different vectors in one-hot representation, we find that $|\ ab\ | = |\ bc\ | = |\ ac\ | = \sqrt{2}$, which means that no matter how semantically similar or different a, b and c are, the distance between them is always $\sqrt{2}$. Therefore, one-hot vector cannot express semantic information. To overcome this problem, we map one-hot vectors to low-dimensional, real-valued and dense vectors.

4.3. Embedding Layer

It is obtained by connecting a fully-connected one above the user input layer and item input layer, respectively. It projects users and items into a k-dimensional (k $\ll$ m, k $\ll$ n) and dense embedding space. The formal representation is as follows:

$$\begin{aligned} e_u &= \mathbf{P_{m*k}}^{\mathsf{T}} \cdot u \\ e_i &= \mathbf{P_{n*k}}^{\mathsf{T}} \cdot i \end{aligned} \tag{3}$$

where $\mathbf{P_{m*k}}^{\mathsf{T}}$ and $\mathbf{P_{n*k}}^{\mathsf{T}}$ are weight matrixes, and they are $u - u$ and $i - i$ embedding layers, respectively. e_u and e_i represent embedding of u and i, respectively. Subsequent training will enable NMetricF to learn the semantic-rich embeddings.

4.4. Metric Interaction Layer + Multi-Layer Perception (MLP)

Since both e_u and e_i are k-dimensional dense vectors, it is easy to calculate the distance between them; that is, subtracting the elements one by one. If only the neurons that represent the difference are squared, it connects to the output one with only one neuron and the square root calculation, which is simplified to the ordinary metric factorization

model (i.e., FML [11]). This simplified form does not introduce a nonlinear hidden layer, so it is essentially a linear regression model similar to matrix factorization. For the sake of obtaining a more accurate result of NmetricF, we use multi-layer MLP, through the gradual dimension reduction of the fully connected layer and nonlinear activation functions, to achieve a more generalized matrix factorization and obtain a more powerful ability to mine complex interactions between users and items. The specific practices are as follows($\ominus$ denotes element-wise subtraction):

$$\begin{aligned} h_0 &= e_u \ominus e_i \\ h_1 &= Relu(\mathbf{W_1}^{\mathsf{T}} h_0 + b_1) \\ &\cdots\cdots \\ h_{L-1} &= Relu(\mathbf{W_{L-1}}^{\mathsf{T}} h_{L-2} + b_{L-1}) \\ \hat{y}_{u,i} &= \mathbf{W_L}^{\mathsf{T}} h_{L-1} + b_L \qquad or \\ \hat{y}_{u,i} &= \sigma(\mathbf{W_L}^{\mathsf{T}} h_{L-1} + b_L) \end{aligned} \tag{4}$$

where W_k ,b_k, and Relu() are standard expressions and meanings. $\hat{y}_{u,i}$ is u's predicted rating (without sigmoid function) or probability of preference (with sigmoid function) on the item i.

4.5. Optimization Method

NMetricF can be applied to various prediction tasks. In the experiment of this paper, we will conduct experiments for two kinds of tasks, respectively: the rating prediction is essentially a regression task, so we use Mean Square Error (MSE) to optimize our model, plus the regular term, so the loss function is:

$$Loss = \frac{1}{\mid D \mid} \sum_{(u,i)\in D} (y_{ui} - \hat{y}_{ui})^2 + \lambda \|W\|^2 \tag{5}$$

where D is entries observed in the metric, $y_{u,i}$ distance between u and i, and $\hat{y}_{u,i}$ is predicted distance. Equation (5) is the objective function that needs to be minimized in the rating prediction task. During the experiments, the Adam algorithm is used to minimize the objective function.

On the other hand, in the ranking recommendation experiment, we choose the common cross-entropy function:

$$Loss = -\frac{1}{\mid D \mid} \sum_{(u,i)\in D} [y_{ui} \ln \hat{y}_{ui} + (1 - y_{ui}) \ln(1 - \hat{y}_{ui})] + \lambda \|W\|^2 \tag{6}$$

5. Experiments

In order to fully test the effectiveness of NMetricF, we perform extensive ablation experiments on two recommended tasks: rating prediction and item ranking.

On each task, we first introduce the experimental setup, including datasets, evaluation metrics, and baseline methods. Then, we will present in detail the comparison of experimental results between our method and the baseline methods. Finally, we analyze and summarize the comparison of these results with the purpose of answering the following questions:

RQ1. Is the NMetricF method superior to baselines on the rating prediction task?

RQ2. How about the generalization ability of NMetricF?

RQ3. Does metric factorization have more advantages than matrix factorization for the task of predicting missing entries of the sparse rating matrix?

RQ4. Does NMetricF outperform baseline methods for item recommendation ranking tasks?

5.1. Evaluation for Rating Prediction

5.1.1. Experiment Settings

Datasets. To perform the rating prediction task, we select the following four real-world datasets for comparison experiments: MovieLens-100K, MovieLens-1M, MovieLens-Latest and Goodbooks. These four datasets can be divided into two categories for presentation.

Goodbooks http://fastml.com/goodbooks-10k/(accessed on 26 January 2022). Gookbooks is widely used in book recommendation, which comes from Goodreads website and contains 5,976,479 ratings of the 10,000 most popular books by 53,424 users.

Movielens https://grouplens.org/datasets/movielens/ (accessed on 26 January 2022). Movielens is a set of datasets collected from movie recommendation websites. There are several versions corresponding to different data volumes, such as Movielens-100K, Movielens-1M, and Movielens-Latest, etc.

The statistical information of all datasets is shown in Table 1:

Table 1. Statistics of four datasets (# denotes "Number of").

Datasets	Users	Items	Ratings	Density
Goodbooks	53,424	10,000	5,976,479	1.12%
Movielens-100K	943	1682	100,000	6.31%
Movielens-1M	6040	3952	1,000,209	4.19%
Movielens-Latest	283,228	193,886	27,753,444	0.05%

Evaluation Metrics. After randomly shuffling all rating entries, the dataset is divided into 80% and 20% for the training and test sets, respectively. During the training of the model, another 2% of the samples from the training set are randomly sampled as the validation set to monitor the overfitting moments.

We employ the widely used metric Root Mean Square Error (RMSE) to assess the error in rating predictions. RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in R}(y_{ui} - \hat{y}_{ui})}{\mid R \mid}} \tag{7}$$

where R represents the set of all user–item rating entries, $y_{u,i}$ is the real rating of the item i by user u, and $\hat{y}_{ui}$ is the predicted rating of the item i by user u.

Baselines. We compare NMetricF with the following competitive rating predicting methods:

- BPMF [13]. BPMF utilizes a Bayesian approach to implement a probability matrix factorization. Automatic control of the model capacity is achieved by integrating all parameters and hyperparameters.
- NRT [14]. NRT uses a gated recurrent network to generate tips to mimic the user's experiences and feelings, which in turn are used to predict ratings.
- NNMF [8]. NNMF approximates the entries of a matrix with a multi-layer feed-forward neural network.
- NCF [9]: NCF uses a neural network architecture to implement matrix factorization. Note that in our paper, we fine-tune the NCF model to predict user–item ratings.
- FML [11]: FML identifies a low-rank structure from the distance factor space and performs recommendation jointly with metric learning methods and factor-based models.

Implementation Details. We run NMetricF's code on a GTX1080 GPU. We use the lecun _uniform distribution to initialize the weight parameters of the model and use Adam/L2 to optimize/regularize the model. For the number of layers of Multi-Layer Perceptron, we set them in order from 1 to 6 for comparison experiments, and a batch normalization layer is added between every two layers instead of the dropout layer. If not specified, the default parameters of the model are set as follows: (1) regularization

coefficient = 0.02, (2) number of MLP layers = 6 (200-150-100-50-25-1), (3) epochs = 200, (4) batch size = 512, (5) learning rate = 0.001 and (6) activation function = 'relu'.

5.1.2. Performance Comparison (RQ1)

In this subsection, we will compare the NMetricF performance with baselines on the rating prediction tasks. Table 2 summarizes the details of all methods. We adopt RMSE to evaluate our experimental rating prediction results. In Table 2, the smaller the value of RMSE, the better the performance of the model. To facilitate the comparison of results across models, for each dataset, the best performance is highlighted in bold, while the second-best performance is underlined. The improvement is calculated as follows. For example, our method achieves a minimum RMSE value of 0.734, while the minimum RMSE value in the baseline method is 0.829, which is the second-best performance. Thus, the performance improvement rate of our method is calculated as follows: $(0.829 - 0.734)/0.829 \times 100\%$, representing the rate of improvement of the best performance over the second-best performance. To summarize, we can draw three observations from the experimental results:

Table 2. RMSE comparison between NMetricF and baseline Models. (Best performance is in boldface and the second-best is underlined.)

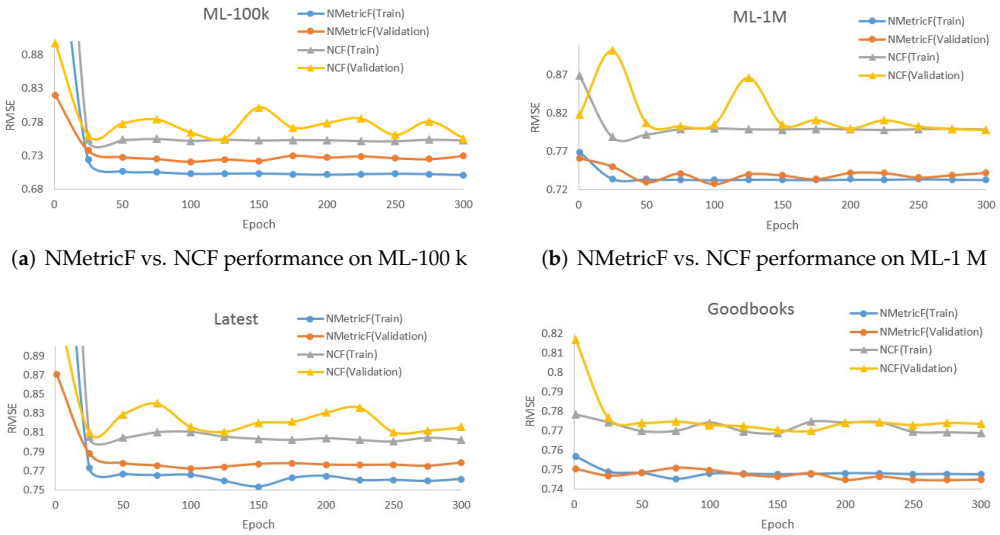
Datasets	**Movielens-100 K**	**Movielens-1 M**	**Movielens-Latest**	**Goodbooks**
Models		**Root Mean Square Error**		
BPMF	0.928	0.869	0.942	0.833
BiasedSVD	0.916	0.839	0.874	0.841
NRT	0.913	0.880	0.869	0.821
NNMF	0.907	0.842	0.869	0.825
NCF	0.902	<u>0.820</u>	0.870	0.814
FML	<u>0.891</u>	0.836	<u>0.855</u>	<u>0.798</u>
NMetricF	**0.738**	**0.731**	**0.742**	**0.747**
Imporve	17.2%	10.9%	13.2%	6.4%

- In general, our NMetricF performs the best among all compared methods. Specifically, on the Movielens-100K dataset, the RMSE value of NMetricF is 0.738, which is 17.2% lower in error compared to the second-best method, FML (0.891); on the Movielens-1M dataset, the RMSE value of NMetricF is 0.731, which is 10.9% lower in error compared to the second-best method, NCF (0.820) which is 10.9% lower in error; on the Movielens-latest dataset, the RMSE value of NMetricF is 0.742, which is 13.2% lower in error compared to the second-best method FML (0.855); on the Goodbooks dataset, the RMSE value of NMetricF is 0.747. On the Goodbooks dataset, the RMSE value of NMetricF is 0.747, which is 6.4% lower than the second-best method, FML (0.798).
- All the experiments in Table 2 can be divided into three categories: BPMF and BiasedSVD are traditional matrix factorization-based models; NRT, NNMF and NCF are neural network-based models; and FML and NMetricF are metric factorization-based models. As can be seen from the table, NRT, NNMF and NCF generally outperform BPMF and BiasedSVD, which proves that the neural network-based model outperforms the matrix factorization-based model, which we believe is due to the inclusion of nonlinear mapping and more hidden layers in the former compared with the latter, thus resulting in the stronger fitting ability of the former. Further, FML and NMetricF generally outperform NRT, NNMF and NCF, which proves that the metric factorization-based model outperforms the neural network-based model, and we believe that this is due to the greater generalization ability of the metric factors used in the former. This will be quantitatively analyzed and explained later in Formula (8).
- Table 2 also shows that the performance of NCF and FML are relatively close, but both are far worse than NMetricF. We believe this is because NCF introduces deep neural networks, but the shortcomings of the dot product still exist; conversely, FML uses metric factorization instead of the dot product, but lacks the deep nonlinear mapping capability of neural networks. It is inspired by this that we propose NMetricF—

which employs metric factorization as an interaction and implements this interaction computation through deep learning techniques—combining exactly the strengths of NCF and FML. As a result, our approach makes substantial progress in performance.

5.1.3. The Advantage of Distance Learning over Dot Product (RQ2, RQ3)

A fundamental difference between our model and NCF is that our model uses metric factorization and NCF uses matrix factorization, so in order to prove the superiority of metric factorization over matrix factorization and to investigate the generalization ability of NMetricF, we compile NMetricF and NCF with exactly the same parameters. The two models have exactly the same hyper-parameters as the following: embedding size = 200, number of nonlinear hidden layers = 6 (200-150-100-50-25-1). Figure 5 shows the error trends of NMetricF and NCF during training on the four data sets.



(**a**) NMetricF vs. NCF performance on ML-100 k

(**b**) NMetricF vs. NCF performance on ML-1 M

(**c**) NMetricF vs. NCF performance on Latest

(**d**) NMetricF vs. NCF performance on ML-1 M

Figure 5. Performance comparison of NMetricF and NCF with the same complexity. (**a**) NMetricF vs. NCF performance on ML-100 k. (**b**) NMetricF vs. NCF performance on ML-1 M. (**c**) NMetricF vs. NCF performance on Latest. (**d**) NMetricF vs. NCF performance on ML-1 M.

From Figure 5 we can see that NMetricF outperforms NCF on both the training and validation sets on each epoch of the training process, provided that the various hyperparameters are identical. This indicates that the distance-based metric factorization is indeed better than the similarity-based matrix factorization. In addition, from Figure 5a–c, we can see that the RMSE values of NCF on the validation set are less stable and oscillate more severely, indicating that NCF shows an overfitting problem. In contrast, NMetrciF only shows a slight overfitting in (a), (c) and almost no overfitting in (b), (d). This indicates that the generalization ability of NMetrciF is excellent. To explain the difference in generalization ability between the two methods, we introduce the error bound formula from computational learning theory:

$$E(h) \leq \hat{E}(h) + O_p(\sqrt{\frac{|H|}{n}}) \tag{8}$$

where $\hat{E}(h)$ is the training error, and $E(h)$ is the validation error. $|H|$ is the size of the model space or hypothesis space. If the model has more parameters or a larger range of parameter values, then the larger $|H|$ is, the more complex the model is. n represents the number of training samples. From Formula (8), we can see that for different models on the same training set (i.e., n is the same), if $|H|$ is larger, $O_p(\sqrt{\frac{|H|}{n}})$ is also larger, which

results in a larger difference between $E(h)$ and $\hat{E}(h)$, i.e., the more severe overfitting. In our opinion, the dot-product operation used for matrix factorization makes the values of neurons entering the MLP less stable, which leads to a larger change in the parameter values of the model during the training process, and eventually leads to an increase in $|H|$. According to Formula (8), this is the reason for the severe overfitting of NCF. On the contrary, the metric calculation used in NMetricF is based on subtraction, and the result must be more stable than that of dot-product, which makes our model have a smaller hypothesis space, i.e., the $|H|$ value is smaller, so the validation error and training error are closer. Therefore, our proposed NMetricF method does not suffer from overfitting problems and has better generalization ability than other methods.

5.2. Evaluation for Item Ranking

5.2.1. Experiment Settings

Datasets. To perform the item ranking task, we select the following two real-world datasets for comparison experiments: FilmTrust and EachMovie.

FilmTrust https://www.librec.net/datasets.html (accessed on 26 January 2022) . FilmTrust is a dataset with trust relationships and movie ratings that were extracted from the FilmTrust website in June 2011 by Guo et al. It contains 35,497 ratings for 2071 movies by 1508 users. Movies are rated ranging from 0.5 to 4 ([0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4]).

EachMovie http://networkrepository.com/rec-eachmovie.php (accessed on 26 January 2022) . EachMovie is provided by the Compaq Systems Research Center. It contains 2,811,717 ratings for 74,424 movies from 1636 users. Movies are rated by six different levels ranging from 1 to 6.

The ranking task in recommendation systems does not focus on the accuracy of predicted ratings but often selects and ranks items based on the probability of users liking them. Therefore, for the above dataset, we first convert the explicit feedback (ratings) to implicit feedback (0 or 1) during preprocessing as follows. For user set $U = \{U_1, U_2, \ldots U_m\}$ and item set $I = \{I_1, I_2, \ldots I_n\}$, we generate $u - i$ interaction matrix Y= $\{y_{ui} \mid u \in \mathrm{U}, \mathrm{i} \in \mathrm{I}\}$ based on the rating information, where

$$y_{ui} = \begin{cases} 1 & \text{if } u \text{ has interacted with } i; \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Note that if u rates i, y_{ui} is 1, representing a positive instance. Conversely, if user u did not rate item i, y_{ui} is set to 0, representing a negative instance. After pre-processing, the statistical characteristics of the two datasets is shown in Table 3:

Table 3. Statistics of four datasets (# denotes "Number of").

Datasets	# Users	# Items	# Ratings	Density
FilmTrust	1508	2071	35,497	1.14%
EachMovie	1636	74,424	2,811,717	2.31%

Evaluation Method. Unlike the practice of rating prediction, the process of ranking recommendation is as follows: for one user, a trained model is used to compute the probability of that user liking all items, and then, based on these probabilities, the items are ranked from largest to smallest, and finally, the top K ones are selected, which are recommended. The method for evaluating this task is often a combination of Precision@K and Recall@K. Precision measures the ratio of the number of items accurately recommended to the total number of recommendations (i.e., K), and recall calculates the ratio of the number of items accurately recommended to the total number of correct items.

Baselines. We compare the performance of NMetricF with the following methods on the item ranking task.

- BPR [15]. BPR performs item ranking based on the maximized posterior probability of Bayesian theory.

- WRMF [16]. WRMF proposes to view implicit feedback such as user behavior as signs of like and detestation for item recommendation.
- NeuMF [9]. NeuMF maps one-hot representations embedding space and then models users' preferences for items by dot-product embeddings. This is a classical and general approach to use neural networks for item recommendation.
- CDAE [10]. CDAE uses an auto-encoder network to generate distributed representations by learning ratings, which in turn predicts the missing items in the rating matrix.
- CML [12]. CML is a collaborative metric learning algorithm; this method extends metric factorization to collaborative filtering.
- FML [11]. FML identifies a low-rank structure from the metric factor space and performs collaborative filtering jointly with metric learning methods and factor-based models.

5.2.2. Performance Comparison (RQ4)

In this subsection, we will compare NMetricF performance with baselines on item ranking task. Table 4 summarizes the results of all methods. We adopt Presicion@K and Recall@K to evaluate our experimental ranking prediction results. In Table 4, the larger the P@K or R@K value, the better the performance of the model. The best performance on each dataset is highlighted in bold, and the second-best one is highlighted with an underline. The improvement is calculated as follows. For example, our method achieves a maximum P@K value of 0.486, while the maximum P@K value in the baseline method is 0.452, which is the second-best performance. Thus, the performance improvement rate of our method is calculated as follows: $(0.486 - 0.452)/0.452 \times 100\%$, representing the rate of improvement of the best performance compared to the second-best performance. From Table 4 we can draw the following conclusions:

Table 4. Comparison between NMetricF and baselines in terms of RMSE. (Best performance is in boldface and second-best is underlined.)

Methods	P@5	P@10	R@5	R@10
FilmTrust				
BPR	0.417	0.346	0.396	0.618
WRMF	0.431	0.347	0.429	0.636
NeuMF	0.415	0.352	0.390	0.629
CDAE	0.431	0.357	0.440	0.652
CML	0.441	<u>0.369</u>	0.438	0.650
FML	<u>0.456</u>	0.361	<u>0.467</u>	<u>0.672</u>
NMetricF	**0.486**	**0.399**	**0.483**	**0.690**
Improve	**6.6%**	**8.1%**	**3.4%**	**2.1%**
EachMovie				
BPR	0.341	0.288	0.307	0.443
WRMF	0.392	0.316	0.347	0.492
NeuMF	0.382	0.307	0.338	0.479
CDAE	0.397	0.313	0.354	0.492
CML	0.396	0.316	0.350	0.491
FML	<u>0.418</u>	<u>0.338</u>	<u>0.368</u>	<u>0.510</u>
NMetricF	**0.425**	**0.351**	**0.379**	**0.531**
Improve	**1.7%**	**3.8%**	**3.0%**	**4.1%**

- NMetricF is the best one among all methods. Specifically, on FilmTrust dataset, compared with the suboptimal method, the performance of NMetricF is improved by 6.6%, 8.1%, 3.4% and 2.1%, respectively, in four different evaluation indicators: p@5, P@10, R@5 and R@10. On the EachMovie dataset, NMetricF outperforms the second method with 1.7%, 3.8%, 3.0%, 4.1%, respectively, in four different evaluation indicators: p@5, P@10, R@5 and R@10. These prove that NMetricF is still superior to baseline methods on the ranking recommendation task.

- Euclidean distance-based models (e.g., NMetricF) outperform dot-product-based ones (e.g., NeuMF), demonstrating that distance interaction is better than dot-product interaction for measuring and characterizing the relationship between the user profile and item profile in the field of the recommender system.

6. Summary and Prospect

Combining metric factorization and deep learning techniques, we propose the neural metric factorization (NMetricF) method. This method first converts the user–item rating matrix into a distance matrix and then fits the distance matrix using the metric interaction of the neural network. Extensive experiments on six datasets demonstrate that NMetricF greatly outperforms previous baseline methods for both rating prediction and item ranking tasks.

NMetricF explores the role of Euclidean distance in recommendations. Experiments show that Euclidean distance interaction can eliminate the incompatibility of triangular inequalities caused by matrix factorization. In addition, large amounts of other structural or auxiliary information indeed exist in real-world scenarios, such as social networks, item contexts or review information. We can add this information to our proposed framework to help with recommendations.

In the following work, we intend to use the neural metric factorization model for a number of related areas in medical research [17–23], and we believe that this model may be useful for applications such as disease prediction.

Author Contributions: Formal analysis, J.Y.; Funding acquisition, P.Z.; Methodology, S.W.; Software, X.S.; Supervision, J.Y.; Writing—original draft, X.S. and L.G.; Writing—review-editing, X.S. and Z.H. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the 2022 Jilin Provincial Education Department Project (Contract No. JJKH20221172KJ), as well as the 2020 Jilin Higher Education Teaching Reform Study and the 2022 Changchun Institute of Humanities and Science Project Bank.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *8*, 30–37. [CrossRef]
2. Koren, Y.; Bell, R. Advances in collaborative filtering. In *Recommender Systems Handbook*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 145–186.
3. Salakhutdinov, R.; Mnih, A. Probabilistic matrix factorization. In Proceedings of the NIPS, Vancouver, BC, Canada, 3–6 December 2007; pp. 1257–1264.
4. Purushotham, S.; Liu, Y.; Kuo, C. Collaborative topic regression with social matrix factorization for recommendation systems. In Proceedings of the ICML, Edinburgh, UK, 26 June–1 July 2012; pp. 759–766.
5. Wang, C.; Blei, D.M. Collaborative topic modeling for recommending scientific articles. In Proceedings of the ACM SIGKDD, KDD '11, San Diego, CA, USA, 21–24 August 2011; pp. 448–456.
6. Lee, J.; Kim, S.; Lebancon, G.; Singeri, Y. Local low-rank matrix approximation. *J. Mach. Learn.* Res. **2016**, *17*, 442–465.
7. Kim, D.; Park, C.; Oh, J. Convolutional matrix factorization for document context-aware recommendation. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; ACM: New York, NY, USA, 2016; pp. 233–240.
8. Dziugaite, G.K.; Roy, D.M. Neural Network Matrix Factorization. *arXiv* **2015**, arXiv:1511.06443.
9. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.-S. Neural collaborative filtering. In *WWW*; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2017; pp. 173–182.
10. Wu, Y.; DuBois, C.; Zheng, A.X.; Ester, M. Collaborative denoising auto-encoders for top-n recommender systems. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, Serie WSDM '16, San Francisco, CA, USA, 22–25 February 2016; ACM: New York, NY, USA, 2016; pp. 153–162.

11. Zhang, S.; Yao, L.; Wu, B.; Xu, X.; Zhang, X.; Zhu, L. Unraveling metric vector spaces with factorization for recommendation. *IEEE Trans. Ind. Inform.* **2020**, *16*, 732–742. [CrossRef]
12. Hsieh, C.K.; Yang, L.; Cui, Y.; Lin, T.Y.; Belongie, S.; Estrin, D. Collaborative metric learning. In Proceedings of the 26th International Conference on World Wide Web, Ser. WWW '17, Perth, Australia, 3–7 May 2017; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2017; pp. 193–201.
13. Salakhutdinov, R.; Mnih, A. Bayesian probabilistic matrix factorization using markov chain monte carlo. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; ACM: New York, NY, USA, 2008; pp. 880–887.
14. Li, P.; Wang, Z.; Ren, Z.; Bing, L.; Lam, W. Neural rating regression with abstractive tips generation for recommendation. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Ser. SIGIR '17, New York, NY, USA, 11–15 July 2017; ACM: New York, NY, USA, 2017; p. 34554.
15. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. Bpr: Bayesian personalized ranking from implicit feedback. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Ser. UAI '09, Montreal, QC, Canada, 18–21 June 2009; AUAI Press: Arlington, VA, USA, 2009; pp. 452–461.
16. Hu, Y.; Koren, Y.; Volinsky, C. Collaborative filtering for implicit feedback datasets. In *ICDM*; IEEE Computer Society: Washington, DC, USA, 2008; pp. 263–272.
17. Chen, W.; Yue, L.; Li, B. DAMTRNN: A delta attention-based multi-task RNN for intention recognition. In *International Conference on Advanced Data Mining and Applications*; Springer: Cham, Switzerland, 2019; pp. 373–388.
18. Chen, W.; Wang, S.; Zhang, X. EEG-based motion intention recognition via multi-task RNNs. In Proceedings of the 2018 SIAM International Conference on Data Mining, San Diego, CA, USA, 3–5 May 2018; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2018; pp. 279–287.
19. Chen, W.; Long, G.; Yao, L. AMRNN: Attended multi-task recurrent neural networks for dynamic illness severity prediction. *World Wide Web* **2020**, *23*, 2753–2770. [CrossRef]
20. Yue, L.; Tian, D.; Chen, W. Deep learning for heterogeneous medical data analysis. *World Wide Web* **2020**, *23*, 2715–2737. [CrossRef]
21. Yue, L.; Shen, H.; Wang, S. Exploring BCI Control in Smart Environments: Intention Recognition Via EEG Representation Enhancement Learning. *ACM Trans. Knowl. Discov. Data TKDD* **2021**, *15*, 1–20. [CrossRef]
22. Zhang, S.; Yao, L.; Huang, C.; Xu, X.; Zhu, L. Position and distance: Recommendation beyond matrix factorization. *arXiv* **2018**, arXiv:1802.04606v1.
23. Guo, G.; Zhang, J.; Yorke-Smith, N. A novel bayesian similarity measure for recommender systems. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI), Beijing, China, 3–9 August 2013; pp. 2619–2625.