Neural Representations in Hybrid Recommender Systems: Prediction versus Regularization

Ramin Raziperchikolaei
ramin.raziperchikola@rakuten.com
Rakuten, Inc.
San Mateo, CA, USA

Tianyu Li
litianyu0315@gmail.com
Rakuten, Inc.
Tokyo, Japan

Young-joo Chung
youngjoo.chung@rakuten.com
Rakuten, Inc.
San Mateo, CA, USA

ABSTRACT

Autoencoder-based hybrid recommender systems have become popular recently because of their ability to learn user and item representations by reconstructing various information sources, including users' feedback on items (e.g., ratings) and side information of users and items (e.g., users' occupation and items' title). However, existing systems still use representations learned by matrix factorization (MF) to predict the rating, while using representations learned by neural networks as the regularizer. In this paper, we define the neural representation for prediction (NRP) framework and apply it to the autoencoder-based recommendation systems. We theoretically analyze how our objective function is related to the previous MF and autoencoder-based methods and explain what it means to use neural representations as the regularizer. We also apply the NRP framework to a direct neural network structure which predicts the ratings without reconstructing the user and item information. We conduct extensive experiments which confirm that neural representations are better for prediction than regularization and show that the NRP framework outperforms the state-of-the-art methods in the prediction task, with less training time and memory.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

hybrid recommender systems, neural networks, regularization

ACM Reference Format:
Ramin Raziperchikolaei, Tianyu Li, and Young-joo Chung. 2021. Neural Representations in Hybrid Recommender Systems: Prediction versus Regularization. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, Online, 5 pages. https://doi.org/10.1145/3404835.3463051

1 INTRODUCTION

The goal of recommender systems is to help users identify the items that best fit their personal tastes from a large set of items [15]. To achieve this goal, recommender systems use different kinds of user and item information. One important source of information is the feedback of users on items, which could be implicit (e.g., click on a link, purchase) [3, 14, 15] or explicit (e.g., a rating between 1 and 5) [8, 15]. On e-commerce platforms, predicting users' explicit feedback (e.g., ratings on reviews) is more desirable because it provides better insight about users' preferences. Therefore, we focus on predicting explicit feedback (i.e., ratings).

We focus on hybrid recommender systems, which use both feedback of the users on items and *side information* to make prediction [1]. Side information of the users and items include the content of items (e.g., category, title, description) and profile of users (e.g., age, location, gender), respectively. Using the feedback and side information jointly helps the hybrid methods to overcome the limitation of approaches that use either feedback or side information, and achieve state-of-the-art results [10, 11].

More specifically, assume we have a sparse rating matrix $\mathbf{R} \in \mathbb{R}^{m\times n}$, where m and n are the number of users and items, respectively, $R_{jk} > 0$ is the rating of the user j on the item k, and $R_{jk} = 0$ means the rating is unknown. Assume the side information of all the users and items are represented by $\mathbf{X}$ and $\mathbf{Y}$, respectively. *The goal of hybrid methods is to predict the unknown ratings using the known ratings and the user and item side information.*

Deep neural networks have become popular in designing hybrid recommendation methods, mainly because of their ability in learning good representations. One of the most widely used neural network structures in recommender systems has been (denoising) autoencoders [2, 9, 10, 16, 18, 20, 21]. These methods define $\mathbf{g}^u()$, $\mathbf{f}^u()$, $\mathbf{g}^i()$, $\mathbf{f}^i()$ as the user's encoder, user's decoder, item's encoder, and item's decoder, respectively. The outputs of the encoders $\mathbf{g}^u()$ and $\mathbf{g}^i()$ are the learned neural representations of the users and items. They also consider $\mathbf{U} \in \mathbb{R}^{m\times d}$ and $\mathbf{V} \in \mathbb{R}^{n\times d}$ as the d-dimensional representations of the users and items, respectively. Their objective function can then be written as:

$$\min_{\mathbf{U},\mathbf{V},\boldsymbol{\theta}} L(\mathbf{f}^u(\mathbf{g}^u(\mathbf{R},\mathbf{X}))) + L(\mathbf{f}^i(\mathbf{g}^i(\mathbf{R},\mathbf{Y}))) + \lambda_1 \sum_{j,k} \mathbb{1}_{R_{jk}} ||R_{jk} - \mathbf{U}_{j,:}\mathbf{V}^T_{k,:}||^2 + \lambda_2 ||\mathbf{U} - \mathbf{g}^u(\mathbf{R},\mathbf{X})||^2 + \lambda_3 ||\mathbf{V} - \mathbf{g}^i(\mathbf{R},\mathbf{Y})||^2 + \text{reg. terms}, \quad (1)$$

where $\boldsymbol{\theta} = [\boldsymbol{\theta}_{f^u}, \boldsymbol{\theta}_{g^u}, \boldsymbol{\theta}_{f^i}, \boldsymbol{\theta}_{g^i}]$ contains all the parameters of the two autoencoders and $\mathbf{U}_{j,:}$ denotes the ith row of the matrix $\mathbf{U}$. The indicator function $\mathbb{1}_{R_{jk}}$ returns 1 when $R_{jk} > 0$, and 0 otherwise. The rating of the user j on item k is approximated by the dot product of the $\mathbf{U}_{j,:}$ and $\mathbf{V}_{k,:}$.

We divide the objective function of Eq. (1) into three parts: 1) the reconstruction losses in the first two terms try to reconstruct the ratings and the side information of the users and items, 2) the MF in the third term decomposes the rating matrix into user and item representations, which will be used for the prediction later, and 3) the fourth and fifth terms try to keep the representations learned by MF in some distance from the neural representations. As we argue in Section. 3, these terms play the role of regularizer, which keep the representations from converging to the solution of MF. The hyper-parameters λ_2 and λ_3 determine how far the two representations should be from each other.

The main issue of this formulation is that the motivation behind using neural representation as the regularizer is unclear. Also, it is difficult to decide how far/close the neural and MF representations should be from each other, i.e., it is difficult to set the hyper-parameters λ_2 and λ_3.

In this paper, we introduce **N**eural **R**epresentation for **P**rediction (**NRP**) framework that learns one set of user and item representations from the neural networks and uses them for the prediction directly, instead of using them as the regularizer. We apply NRP to the autoencoder structure, analyze its objective function and optimal solution, and compare it with the previous approaches based on autoencoder and MF. We also introduce a direct neural network structure integrated with our framework to obtain more expressive power and training efficiency. Our experiments show that 1) neural representations perform better in prediction than regularization and 2) the direct neural network structure outperforms previous methods, while having faster training and less memory usage.

2 RELATED WORK

Collaborative filtering (CF) methods learn users' preferences from the patterns in the past interactions between the users and items. Matrix factorization (MF), a well-known CF method, decomposes the rating matrix into two low-rank user and item matrices (representations), such that the dot product of the representations approximates the rating matrix [7, 8, 19]. Since the CF methods only uses the past user-item interactions, their performance degrades significantly when the rating matrix is highly sparse or when we have new users and items in the system (cold-start problem) [15]. Hybrid methods address this issue by using the past interactions and the side information jointly to predict the future ratings. The autoencoder-based hybrid methods apply the autoecnoder structure to extract features from side information and combine such features with the feedback pattern to predict ratings [2, 9, 10, 18, 20, 21]. Recent works [2, 9, 10] utilize side information and ratings of users/items by training two autoencoders. As explained in the introduction, these methods use MF's representations for the prediction. There are also cases where the MF and neural representations are mixed for prediction. In CDL [20] and AutoSVD [21], the item's representation comes from the autoencoder and the user's representation comes from the MF. These methods use the side information of the items as the only source of information, and use dot product to combine the representations. In our method, both users' and items' representations are generated from the neural network, where the inputs are the user/item rating vectors and side information.

3 OUR PROPOSED METHOD

Our main idea is to remove the MF terms and use the neural representations of the users and items for the prediction task. For concision, we call our method **NRP**, which stands for **N**eural **R**epresentation for **P**rediction. We first apply our framework to the autoencoder structure and show how it is related to the previous autoencoder-based methods. Then, we integrate this framework with a direct neural network structure.

3.1 NRP with autoencoders

Similar to the previous works, our model contains two autoencoders, one for the users and one for the items. The difference is that the encoders' outputs are the only user/item representations in our model. Here is our objective function:

$$\min_{\boldsymbol{\theta}} L(\mathbf{f}^u(\mathbf{g}^u(\mathbf{R},\mathbf{X}))) + L(\mathbf{f}^i(\mathbf{g}^i(\mathbf{R},\mathbf{Y})))+ \quad (2)$$
$$\lambda_1 \sum_{j,k} \mathbb{1}(R_{jk} > 0)||R_{jk} - \mathbf{g}^u(\mathbf{R}_{j,:},\mathbf{X}_{j,:})^T \mathbf{g}^i(\mathbf{R}_{:,k},\mathbf{Y}_{k,:})||^2.$$

To have an apple-to-apple comparison between our approach and the previous ones, we use the same loss functions and the decoder and encoder structures as aSDAE [2] and DHA [10].

Our objective in Eq. (2) gives three advantages over the previous works: 1) the hyper-parameters λ_2 and λ_3, from Eq. (1), are removed, 2) the number of parameters decreased as we removed **U** and **V**, which helps in faster training and saving memory, and 3) the network can be trained end-to-end, as there is no need to optimize over **U** and **V**. In Section 4. we will see that these advantages lead to better prediction performance.

We now analyze the objective function of Eq. (2), compare it with the one in Eq. (1), and explain why the neural representations act as a regularizer in previous works. First, we rewrite our objective in (2) as follows:

$$\min_{\boldsymbol{\theta},\mathbf{U},\mathbf{V}} Q(\boldsymbol{\theta},\mathbf{U},\mathbf{V}) = L(\mathbf{f}^u(\mathbf{g}^u(\mathbf{R},\mathbf{X}))) + L(\mathbf{f}^i(\mathbf{g}^i(\mathbf{R},\mathbf{Y})))+$$
$$\lambda_1 \sum_{j,k} \mathbb{1}_{R_{jk}}||R_{jk} - \mathbf{U}_{j,:}^T\mathbf{V}_{k,:}||^2 \text{ s.t. } \mathbf{U} = \mathbf{g}^u(\mathbf{R},\mathbf{X}),\ \mathbf{V} = \mathbf{g}^i(\mathbf{R},\mathbf{Y}). \quad (3)$$

The objective functions in Equations (2) and (3) are equivalent, so we focus on comparing (3) with (1).

We consider two special cases of the objective function in Eq. (1). First, consider the case where $\lambda_2 = \lambda_3 = 0$, which makes the last two terms 0. The first two terms can also be removed since they do not contain the user/item representations **U** and **V**. So only the MF term remains. The second case is when $\lambda_2 = \lambda_3 \to \infty$. The following theorem shows that in this case the two objective functions in (1) and (3) will be equivalent (i.e. they have the same optimal solution).

Theorem 3.1. *The objective function of Eq.* (1), *with* $\lambda_2 = \lambda_3 \to \infty$, *has the same optimal solution as the objective function of Eq.* (3).

Proof. We define a vector $\mathbf{p} = [\boldsymbol{\theta},\mathbf{U},\mathbf{V}]$, containing all the parameters of the problem. Note that $Q(\mathbf{p})$, defined in Eq. (3), contains the first three terms of Eq. (1) and contains all the terms of Eq. (3). We assume that $\bar{\mathbf{p}} = [\bar{\boldsymbol{\theta}},\bar{\mathbf{U}},\bar{\mathbf{V}}]$ is the optimal solution of Eq. (3) and $\mathbf{p}^* = [\boldsymbol{\theta}^*,\mathbf{U}^*,\mathbf{V}^*]$ is the optimal solution of Eq. (1) when $\lambda_2 = \lambda_3 \to \infty$. We replace $\mathbf{p}^*$ and $\bar{\mathbf{p}}$ in Eq. (1) to get the following inequality:

$$\lim_{\lambda_2\to\infty} Q(\mathbf{p}^*)+\lambda_2||\mathbf{U}^*-\mathbf{g}^u(\mathbf{R},\mathbf{X};\boldsymbol{\theta}^*_{g^u})||^2+\lambda_3||\mathbf{V}^*-\mathbf{g}^i(\mathbf{R},\mathbf{Y};\boldsymbol{\theta}^*_{g^i})||^2 \leq$$
$$Q(\bar{\mathbf{p}}) + \lambda_2(||\bar{\mathbf{U}} - \mathbf{g}^u(\mathbf{R},\mathbf{X};\bar{\boldsymbol{\theta}}_{g^u})||^2 + ||\bar{\mathbf{V}} - \mathbf{g}^i(\mathbf{R},\mathbf{Y},\bar{\boldsymbol{\theta}}_{g^i})||^2) = Q(\bar{\mathbf{p}}) \quad (4)$$

The right side of the above inequality is $Q(\bar{\mathbf{p}})$ since $\bar{\mathbf{p}}$ is the optimal solution of Eq. (3) and satisfies the constraints. By rearranging the last equation we obtain

$$||\mathbf{U}^* - \mathbf{g}^u(\mathbf{R}, \mathbf{X}; \boldsymbol{\theta}^*_{g^u})||^2 + ||\mathbf{V}^* - \mathbf{g}^i(\mathbf{R}, \mathbf{Y}, \boldsymbol{\theta}^*_{g^i})||^2 \leq \lim_{\lambda_2 \to \infty} \frac{Q(\bar{\mathbf{p}}) - Q(\mathbf{p}^*)}{\lambda_2} = 0. \quad (5)$$

To satisfy the inequality, i.e., making the sum of the norms 0, both norms have to be 0. This means that $\mathbf{p}^*$ is a feasible vector, i.e. $\mathbf{U}^* = \mathbf{g}^u(\mathbf{R}, \mathbf{X}; \boldsymbol{\theta}^*_{g^u})$ and $\mathbf{V}^* = \mathbf{g}^i(\mathbf{R}, \mathbf{Y}, \boldsymbol{\theta}^*_{g^i})$. By considering feasibility of $\mathbf{p}^*$ in Eq. (4) (i.e., replacing $\mathbf{U}^*$ by $\mathbf{g}^u(\mathbf{R}, \mathbf{X}; \boldsymbol{\theta}^*_{g^u})$ and $\mathbf{V}^*$ by $\mathbf{g}^i(\mathbf{R}, \mathbf{Y}, \boldsymbol{\theta}^*_{g^i})$ in Eq. (4)), we get $Q(\mathbf{p}^*) \leq Q(\bar{\mathbf{p}})$. Note that we assumed that $\bar{\mathbf{p}}$ is the optimal solution of Eq. (3), so for two feasible points $\bar{\mathbf{p}}$ and $\mathbf{p}^*$ we have $Q(\mathbf{p}^*) \geq Q(\bar{\mathbf{p}})$. So the conclusion is that $Q(\mathbf{p}^*) = Q(\bar{\mathbf{p}})$. In other words, for $\lambda_2 = \lambda_3 \to \infty$, the objective functions of the Eq. (1) and Eq. (3) become equivalent. □

By setting $\lambda_2 = \lambda_3 = 0$ and increasing it to $\lambda_2 = \lambda_3 \to \infty$, a path of solutions will be created, between the solution of the MF and our NRP autoencoder. The previous autoencoder methods use a fixed $\lambda_2 > 0$ and $\lambda_3 > 0$, so their optimal solution lies somewhere on the path. The smaller (larger) these hyper-parameters, the closer (farther away) the solution of the autoencoder-based methods will be to the MF's solution. *From this analysis, we believe the neural representations act as the regularizer in previous works since they are only used to keep* **U** *and* **V** *away from the MF's optimal solution.*

A question arises here: can we optimize the objective of Eq. (1) with $\lambda_2 = \lambda_3 \to \infty$ and expect to get the same result as our objective function in Eq. (2)? In practice, this will not happen for two reasons. First, as we set $\lambda_2 = \lambda_3 \to \infty$, the Hessian of the objective function in Eq. (1) becomes ill-conditioned and the first-order methods iterate zigzag and slowly approach toward the optimal solution. The reason is that some eigenvalues of the Hessian will be in the order of ∞, while the other ones will be small. More information can be found in Chapter 17.1 of [12]. Second, note that we have proved the equivalency of Eq. (1) and (2) in terms of their optimal solutions. In practice, the objective functions are highly non-convex and we can expect the methods to end up close to a local solution.

3.2 NRP with a direct structure

We define the direct structure as a neural network structure without the decoders, which predict the ratings without reconstructing the user/item ratings and side information. Our goal of designing the direct structure and combining it with NRP framework is to use it as a baseline. The comparison between the direct structure and the autoencoder-based methods let us know the effectiveness of the reconstruction based methods in hybrid recommendation systems.

Our direct structure is achieved by making two modifications to our autoencoder structure. First, we remove the decoders from the structure, which leads to saving around 50% of memory and faster optimization. Second, we use a set of fully connected layers to predict the final rating, instead of the dot product. This makes our model more expressive with a relatively small number of additional parameters. In the following, we detail how to learn the representations and learn the ratings with the direct structure.

Learning the user and item representations. We focus on the user representation, as the item representation can be learned in the same way. One source of information is the ratings of the users on items. For the jth user, $\mathbf{R}_{j,:}$ contains the ratings of this user on all items. We give this as the input to a set of fully connected layers and get the rating representation, denoted by $\mathbf{g}^u_{\mathrm{r}}(\mathbf{R}_{j,:})$.

The side information of the jth user, denoted by $\mathbf{X}_{j,:}$, will be given as the input to another neural network, which maps it to a low-dimensional representation, denoted by $\mathbf{g}^u_{\mathrm{s}}(\mathbf{X}_{j,:})$.

The final representation for the jth user is achieved by concatenating the two representations above. We show this by $\mathbf{z}_j = [\mathbf{g}^u_{\mathrm{r}}(\mathbf{R}_{j,:}), \mathbf{g}^u_{\mathrm{s}}(\mathbf{X}_{j,:})]$, where $[\quad]$ is used for concatenation of the vectors. We can get the kth item representation $\mathbf{z}_k$ in a similar way, using item interaction vector and item side information $\mathbf{Y}$.

Learning the rating. We concatenate the user and item representations to get the joint representation $\mathbf{z}_{jk}$, which is connected to a neural network to predict the final rating. We show this last neural network by $h()$. We define the objective function as the mean squared error between the output of $h()$ and the true rating:

$$\frac{1}{mn} \min_{\theta} \sum_{j=1}^{m} \sum_{k=1}^{n} \mathbb{1}(R_{jk} > 0)||R_{jk} - h(\mathbf{z}_{jk})||^2 + \lambda_1 ||\boldsymbol{\theta}||^2 \quad \text{s.t.}$$

$$\mathbf{z}_{jk} = [\mathbf{z}_j, \mathbf{z}_k],\ \mathbf{z}_j = [\mathbf{g}^u_{\mathrm{r}}(\mathbf{R}_{j,:}), \mathbf{g}^u_{\mathrm{s}}(\mathbf{X}_{j,:})], \mathbf{z}_k = [\mathbf{g}^i_{\mathrm{r}}(\mathbf{R}_{:,k}), \mathbf{g}^i_{\mathrm{r}}(\mathbf{Y}_{k,:})].$$

We jointly optimize all parameters of the neural network using stochastic gradient descent. Note that our method works as long as we have at least one source of information for users and one source of information for items.

4 EXPERIMENTS

For each dataset, we randomly select 80% of the ratings as the training set, 10% as the validation set, and 10% as the test set. We repeat the process three times to create three training/validation/test sets and report the mean and standard deviation of each method on these three sets. Unless otherwise stated, we use bag of words (BoW) to represent the item and user side information.

Datasets. We use three datasets in our experiments. The ml100k [4] contains 100 000 ratings (1 to 5) from around 1 000 users on 1 600 movies. The user side information contains age, gender, occupation, and zip code and the item side information contains the movie title and the genre. The ml1m [4] dataset contains 1 million ratings of around 6 000 users on 4 000 movies. The content of the user and item side information are the same as ml100k. Ichiba dataset contains 1.5 million Rakuten Ichiba[1]'s product review from 324 000 users on 294 000 items. The user side information contains age and gender and the item side information contains the category.

Evaluation metrics. We report the root mean square error (RMSE) and average precision to evaluate the rating prediction performance. In RMSE we measure the error between the predicted and actual rating. In precision, for each user, we retrieve the top p% of the items with the highest predicted ratings and compare them with the actual top p% of items. We compute the precision for each user and report the average score.

Additional experiments and experimental settings. The extended version of this paper[13] contains additional experimental results, including precision on multiple datasets, and contains experimental

[1]https://rit.rakuten.co.jp/data_release/

Table 1: **Our NRP framework achieves better prediction results, faster training, and less memory usage compared to the autoencoder-based methods. The first/second number in the fourth column is the number of parameters involved in learning the neural network's/MF's representations. Time refers to the training time per epoch.**

	ml1m dataset			
method	RMSE	precision	# params.	time
MF	0.892 ± 0.004	$68.2\% \pm 0.3$	(0, 1M)	45s
DHA	0.865 ± 0.001	$69.3\% \pm 0.2$	(44M, 1M)	1 097s
$\mathbf{NRP_{DHA}}$	0.855 ± 0.002	$69.6\% \pm 0.2$	(44M, 0)	1 027s
aSDAE	0.879 ± 0.005	$69.0\% \pm 0.1$	(66M, 1M)	1 155s
$\mathbf{NRP_{aSDAE}}$	0.877 ± 0.008	$68.5\% \pm 0.4$	(66M, 0)	1 055s
ACCM	0.856 ± 0.002	$69.5\% \pm 0.3$	(11.5M, 0)	450s
$\mathrm{ACCM_{MLP}}$	0.865 ± 0.002	$68.9\% \pm 0.2$	(11.8M, 0)	470s
$\mathbf{NRP_{direct}}$	$\mathbf{0.851 \pm 0.001}$	$\mathbf{70.0\% \pm 0.1}$	(22M, 0)	640s

settings, including all the hyper-parameters in the experiments. The source code is available at https://rit.rakuten.co.jp/oss/.

Neural representations are better in prediction than regularization. In Table 1, we compare our proposed method with matrix factorization (MF), the autoencoder-based methods, DHA [10] and aSDAE [2], and the attention-based direct structure, ACCM [17], on the ml1m dataset. We also replaced the dot product of ACCM with MLP, denoted by $\mathrm{ACCM_{MLP}}$ in Table 1. Our framework applied to the same encoder-decoder structure as DHA and aSDAE are called $\mathrm{NRP_{DHA}}$ and $\mathrm{NRP_{aSDAE}}$, respectively. Our framework applied to the direct structure is called $\mathrm{NRP_{direct}}$.

MF has the worst performance (the largest RMSE), which means that MF formulation with the L2 norm of weights as the regularization is not enough for the rating prediction. By setting the hyper-parameters carefully, both DHA and aSDAE outperform MF, which suggests that the neural network's representations are better regularizers than the L2 norm of the weights. Our methods $\mathrm{NRP_{DHA}}$ and $\mathrm{NRP_{aSDAE}}$ outperform the original DHA and aSDAE, respectively, in terms of RMSE and precision. This improvement comes from removing the MF terms, relying on neural representations, and staying inside the feasible set of neural network's output. ACCM, the direct structure of Shi et al. [17], outperforms MF, DHA, and aSDAE, but shows similar performance to the $\mathrm{NRP_{DHA}}$. Finally, $\mathrm{NRP_{direct}}$ has the best performance and outperforms all autoencoder-based methods. This shows that removing the decoders, making the neural network free of reconstructing the inputs, and replacing the dot product with MLPs lead to learning a better model.

Note that $\mathrm{NRP_{direct}}$ outperforms ACCM because of 1) using interaction vectors as the input instead of IDs and 2) using MLPs, instead of the dot product, to map the joint representation to the rating. To show the importance of using interaction vector as the input, we have replaced the dot product of ACCM with MLPs, denoted by $\mathrm{ACCM_{MLP}}$ in Table 1. $\mathrm{ACCM_{MLP}}$ performs slightly worse than ACCM in ml1m. $\mathrm{NRP_{direct}}$ outperforms $\mathrm{ACCM_{MLP}}$.

The last two columns of Table 1 compare the number of learnable parameters and training time per epoch of each method. DHA and aSDAE have the largest memory usage and training time, as they optimize the two representations alternatively. $\mathrm{NRP_{DHA}}$ and $\mathrm{NRP_{aSDAE}}$ achieve better training time and memory than DHA and aSDAE, respectively. Among the neural network-based methods, the direct structures, ACCM and $\mathrm{NRP_{direct}}$, have the fastest training and lowest memory usage because of their simple structure. Note that $\mathrm{NRP_{direct}}$ has a larger number of parameters than ACCM. This is because $\mathrm{NRP_{direct}}$ uses MLPs to learn low-dimensional representations, while ACCM uses embedding layers.

Table 2: RMSE of our NRP framework compared with the hybrid and collaborative filtering methods. Our approach outperforms the rest of the methods.

method	ml100k	ml1m	Ichiba
MF [8]	0.940 ± 0.003	0.892 ± 0.0004	1.00 ± 0.104
Autorec [16]	0.921 ± 0.002	0.889 ± 0.0003	2.47 ± 0.059
NeuMF [5]	0.948 ± 0.005	0.886 ± 0.001	0.900 ± 0.004
DSSM [6]	0.934 ± 0.002	0.941 ± 0.0004	0.913 ± 0.003
DHA [10]	0.939 ± 0.002	0.865 ± 0.001	OM
$\mathrm{NRP_{DHA}}$	0.926 ± 0.004	0.855 ± 0.002	OM
aSDAE [2]	0.946 ± 0.005	0.879 ± 0.005	OM
$\mathrm{NRP_{aSDAE}}$	0.910 ± 0.008	0.877 ± 0.008	OM
HIRE [11]	0.930 ± 0.006	0.861 ± 0.004	OM
$\mathrm{NRP_{direct}}$	$\mathbf{0.897 \pm 0.003}$	$\mathbf{0.851 \pm 0.001}$	$\mathbf{0.889 \pm 0.002}$

Comparison with other hybrid and collaborative filtering methods. We compare RMSE of our method with several baselines and recent works in Table 2. We slightly modified NeuMF [5] and DSSM [6] to make them work with explicit feedback. We can see that our method achieves the best results on different datasets.

We got an out-of-memory (OOM) error in the training of HIRE, DHA, and aSDAE in Ichiba dataset. HIRE's code (publicly available by the authors) needs the entire interaction matrix in training, which can not be stored in memory for the Ichiba datasets. DHA and aSDAE reconstruct the users' and items' interaction vectors. This makes the first and the last layer of their autoencoders huge in the Ichiba dataset and makes their model's size larger than the $12GB$ memory of our GPU.

5 CONCLUSION

The current autoencoder-based hybrid recommender systems learn matrix factorization-based representations for the prediction task and neural networks-based representation for the regularization. In this paper, we proposed a new framework that uses the neural networks' representation directly for the prediction task. We showed that by applying our approach to the same autoencoder structure as previous works, we can achieve faster training and better performance. We also proposed a new framework by removing the decoders and replacing dot product with MLP in autoencoders. Our approach combined with the proposed framework outperformed the previous works. It also had a fast training and small memory usage compared to the autoencoder-based methods.

REFERENCES

[1] Robin Burke. 2007. Hybrid Web Recommender Systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*. Springer Berlin Heidelberg, 377–408. https://doi.org/10.1007/978-3-540-72079-9_12

[2] Xin Dong, Lei Yu, Zhonghuo Wu, Yuxia Sun, Lingfeng Yuan, and Fangxi Zhang. 2017. A hybrid collaborative filtering model with deep structure for recommender systems. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[3] Zhi-Hong Dong, Ling Huang, Chang-Dong Wang, Jian-Huang Lai, and Philip S Yu. 2019. DeepCF: A Unified Framework of Representation Learning and Matching Function Learning in Recommender System. In *AAAI Conference on Artificial Intelligence*.

[4] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (dec 2015), 1–19. https://doi.org/10.1145/2827872

[5] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[6] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *International conference on Conference on information & knowledge management(CIKM)*. ACM Press. https://doi.org/10.1145/2505515.2505665

[7] Yehuda Koren. 2008. Factorization meets the neighborhood:a Multifaceted Collaborative Filtering Model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD*. ACM Press. https://doi.org/10.1145/1401890.1401944

[8] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (aug 2009), 30–37. https://doi.org/10.1109/mc.2009.263

[9] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management - CIKM*. ACM Press. https://doi.org/10.1145/2806416.2806527

[10] Tianyu Li, Yukun Ma, Jiu Xu, Bjorn Stenger, Chen Liu, and Yu Hirate. 2018. Deep Heterogeneous Autoencoders for Collaborative Filtering. In *IEEE International Conference on Data Mining (ICDM)*.

[11] Tianqiao Liu, Zhiwei Wang, Jiliang Tang, Songfan Yang, Gale Yan Huang, and Zitao Liu. 2019. Recommender Systems with Heterogeneous Side Information. In *The World Wide Web Conference on (WWW)*. ACM Press. https://doi.org/10.1145/3308558.3313580

[12] Jorge Nocedal and Stephen Wright. 2006. *Numerical Optimization*. Springer.

[13] Ramin Raziperchikolaei, Tianyu Li, and Young joo Chung. 2020. Neural Representations in Hybrid Recommender Systems: Prediction versus Regularization. arXiv:2010.06070 [cs.IR]

[14] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*.

[15] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). 2011. *Recommender Systems Handbook*. Springer US. https://doi.org/10.1007/978-0-387-85820-3

[16] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM Press. https://doi.org/10.1145/2740908.2742726

[17] Shaoyun Shi, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Attention-based Adaptive Model to Unify Warm and Cold Starts Recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*. ACM. https://doi.org/10.1145/3269206.3271710

[18] Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid Recommender System based on Autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems - DLRS 2016*. ACM Press. https://doi.org/10.1145/2988450.2988456

[19] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2008. Matrix factorization and neighbor based algorithms for the netflix prize problem. In *Proceedings of the 2008 ACM conference on Recommender systems - RecSys*. ACM Press. https://doi.org/10.1145/1454008.1454049

[20] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM Press. https://doi.org/10.1145/2783258.2783273

[21] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. AutoSVD++:An Efficient Hybrid Collaborative Filtering Model via Contractive Auto-encoders. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press. https://doi.org/10.1145/3077136.3080689