Hybrid CF on Modeling Feature Importance with Joint Denoising AutoEncoder and SVD++

Qing Yang[1], Heyong Li[1], Ya Zhou[2], Jingwei Zhang[2(✉)], and Stelios Fuentes[3]

[1] Guangxi Key Laboratory of Automatic Detection Technology and Instrument, Guilin University of Electronic Technology, Guilin 541004, China
[2] Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China
gtzjw@hotmail.com
[3] Leicester University, Leicester, UK

Abstract. AutoEncoder is an unsupervised learning approach that can maps inputs to useful intermediate features, which can be used to build recommendation. Intermediate features of different entities obtained by AutoEncoder may have different weight for predicting users behavior. However, existing research typically uses a uniform weight on intermediate features to make a fast learning algorithm, this general approach may lead to the limited performance of the model. In this paper, we proposes a novel approach by using SGD to dynamically learn the intermediate features importance, which can integrate the intermediate features into matrix factorization framework seamlessly. In the previous works, the entities intermediate features learned by AutoEncoder are modeled as a whole. On this basis, we proposes to use attention parameters in entity intermediate feature to dynamically learn the intermediate features importance and build fine-grained model. By learning unique attention unit for each entity intermediate feature, the entities intermediate features are integrated into the matrix factorization framework better. Extensive experiments conducted over two real-world datasets demonstrate our proposed approach outperforms the compared models.

Keywords: Denoising AutoEncoder · Collaborative Filtering · Attention unit

1 Introduction

With the explosive growth of online information, the recommendation system is gradually widely deployed in various terminal devices to meet the user's information screening needs [22]. Among the various recommendation strategies, Collaborative Filtering (CF) has been widely adopted due to its precision and efficiency [21]. However, the existing CF methods are insufficient to model nonlinear relation and side information of user and item entities [17,20]. With the widespread

H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 349, pp. 290–304, 2021.
https://doi.org/10.1007/978-3-030-67537-0_18

application of deep learning, many studies apply neural networks to alleviate this problem [28].

Based on CF, [5] present a general framework named NCF, which use neural network to model latent features of users and items, on this basis, [1] adopt an attention mechanism to adapt the representation of a group, and recommendation items for group. [27] propose a dual channel hypergraph collaborative filtering (DHCF) method to model the representation of users and items so that these two types of data can be seamlessly interconnected while still keeping their specific properties unchanged. [11] proposed a new method named Field-weighted Factorization Machines (FwFMs) to model the different feature interactions between different fields in a much more memory-efficient way. [2] describe a novel Field-Leveraged embedding network to learn inter-field and intra-field feature interactions.

However, many existing works compute the feature interactions in a simple way and care less about the importance of features. Due to this, [10] propose AutoFIS to automatically identify important feature interactions for factorization models by training the target model to convergence. [25] propose an automated interaction discovering model for CTR prediction named AutoCTR. [7] propose a new model named FiBiNET, aim to model feature importance by using the the Squeeze-Excitation network (SENET) mechanism. [13] devise AutoInt to automatically learn the high-order feature interactions of input features, these models have made great progress.

Recent research have demonstrated that AutoEncoder have the capability of capturing the complex relationships within raw data, and compact representations in the hidden layers [16]. Based on this, many excellent recommendation models have been proposed [12,18,23]. These research can be divided into two categories [24]. The first category focuses on designing recommendation models based on AutoEncoder only, without using any components of traditional recommendation models. For example, [18] applies the Denoising AutoEncoder to model distributed representations of the users and items via formulating the user-item feedback data. [12] formulated Collaborative Filtering as a AutoEncoder. However, these methods do not employ any extra information of users and items, which leads to the second research category. It aims to use AutoEncoder to learn intermediate feature representations and embed them into classic CF models. For instance, [23] improve Collaborative Filtering (CF) by integrating intermediate features of item entities into matrix factorization framework. The intermediate features are obtained by AutoEncoder. [23] has achieved a good feedback. However, it unified the intermediate features weight to model, which may lead to limited improvement of model performance. In fact, not every dimension of the intermediate features of different item entities obtained by unsupervised approach has predictability [3,19]. Some feature elements contribute little to the prediction, and the useless feature elements even introduce noise, which can result in model to be hindered [6].

In this paper, we design a multi-dimensional attention parameter that seamlessly fuse the intermediate features of the item entity with the factorization

framework, it can be learned by stochastic gradient descent. Compared with the previous research in this direction, the work of our paper is summarized as follows:

- We propose to use learn-able attention parameter to discriminate the importance of each dim of intermediate feature learned from AutoEncoder. More meaningful, the weight of intermediate feature is learned by stochastic gradient descent automatically.
- We conduct extensive experiments on two real-world datasets, and the results show that our model outperforms the compared methods significantly.

The rest of the paper is organized as follows. Section 2 provides the problem definition and revisits SVD-based latent factor models and AutoEncoder framework. Section 3 describes our method and learning algorithm in detail. Section 4.1 presents experimental results for the performance comparisons and components analysis. Section 5 summarizes our work and discusses our future directions.

2 Preliminaries

2.1 Problem Definition

Given a set of users $U = [1, ..., M]$, a set of items $I = [1, ..., N]$, we use r_{ui} to express the rating of the user u for the item i, and $r_{ui} \subset R \in \mathcal{R}^{M \times N}$, R is an incomplete matrix of ratings [14]. The known score is expressed as $I = \{(u, i)|r_{ui} \ is \ known\}$. We use $\hat{r_{ui}}$ represents the predicted value of r_{ui}, which captures the interaction score between the user u and the item i.

2.2 Denoising AutoEncoder

A traditional AutoEncoder takes a vector $x \in [0, 1]^d$ as input, and transforms it into hidden representation $y \in [0, 1]^{\acute{d}}$ through a deterministic mapping:

$$y = h_\theta(x) = s(W \cdot x + b) \tag{1}$$

Its parameter set is $\Theta = \{W, b\}$, where W is a $\acute{d} \times d$ weights matrix and b is a biases column vector. The hidden representation y is then mapped back to a reconstructed vector $z \in [0, 1]^d$ through:

$$z = h_\theta^{'}(y) = s(W^{'} \cdot y + b^{'}) \tag{2}$$

With parameters set $\Theta^{'} = \{W^{'}, b^{'}\}$. The weight matrix $W^{'}$ of reverse mapping may be constrained by $W^{'} = W$ in an optional manner. The parameters of this model are determined by minimize the average reconstruction error:

$$\arg\min_{\Theta, \Theta'} \frac{1}{n} \sum_{i=1}^{n} L(x, z) \tag{3}$$

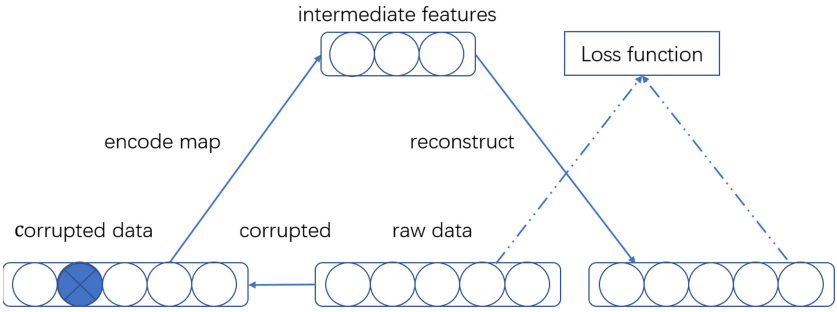

Fig. 1. The architecture of Denoising AutoEncoder

Where L is the traditional square error loss function $L(x-z) = \|x-z\|^2$.

As shown in see Fig. 1, the Denoising AutoEncoder [15] reconstruct a repaired input x from a corrupted version $\widetilde{x}$ by virtue of a stochastic mapping $\widetilde{x} \sim p(\widetilde{x}|x)$. There are two common corruption choices include additive Gaussian noise and the multiplicative mask-out/drop-out noise [18]. In this paper, we choose Gaussian noise to corrupt input data.

2.3 Latent Factor CF Models

Funk-SVD. Funk-SVD [4]is a classic CF model, which map features of users and items to a shared latent factor space of dimensionality k, each item i corresponds to a latent property vector $q_i \in \mathbb{R}^k$, and each user u corresponds to a hidden preference vector $p_u \in \mathbb{R}^k$. The prediction is done by taking an inner product $\hat{r_{ui}} = q_i^T \cdot p_u$, which denote the prediction score of user u to item i.

Biased SVD. It would be inadequate to explain the full rating value by latent factor interaction. Based on Funk-SVD, [8] propose a improved version named biased SVD, which introduces the concept of user and item deviation, the prediction equation is as follows:

$$\hat{r_{ui}} = \mu + b_u + b_i + q_i^T \cdot p_u \tag{4}$$

The parameter μ denotes the global average rating of each user, b_i indicates the bias term of item i, b_u is observed bias of user u, q_i and p_u inherit the definition of Funk-SVD.

SVD++. When explicit feedback is insufficient, implicit information such as browsing and purchase history information can be used to gain insight into user preferences and alleviate the cold start problem. SVD++ [8]is an improved model by minding the biased SVD. The model is more precise, the prediction formula is as follows:

$$\hat{r_{ui}} = \mu + b_u + b_i + q_i^T \cdot (p_u + |N(u)|^{-\frac{1}{2}} \cdot \sum_{j \in N(u)} y_j) \tag{5}$$

Where $N(u)$ denotes the items for which user u uttered an implicit preference, y_j represents the latent vector of theses item. The $|N(u)|^{-\frac{1}{2}} \cdot \sum_{j \in N(u)} y_j$ in the Eq. (5) is used to enhance user implicit representation.

3 Proposed Methodology

In this section, we will introduce our approach named HCF-DAE, which can distinguish the importance of intermediate features of the item by using attention parameters. In this way, the intermediate features of the item can be better integrated into SVD++.

3.1 Our Approach HCF-DAE

Based on autoencoder, [23] applies the item entity intermediate features to the SVD++ framework, constructs hybrid model named Auto-SVD++, and improves prediction performance. However, the Auto-SVD++ embed item entity intermediate features into SVD++ with a uniform weights, which can limit performance of model. For example, for user u, the intermediate features $CAE(i)$ of item entity i contributes to model performance improvement, we name it a useful feature, however, the intermediate features $CAE(j)$ of item entity j may be useless features, and even affect the predictive performance of the model. Based on the above analysis and [23], our paper proposes a new method named HCF-DAE, the architecture of our proposed model see Fig. 2, where p_u and q_i represent the latent features of users and items respectively. Attention unit in Fig. 2 is equivalent to our attention strategy, which can distinguish the importance of item entity intermediate features learn from DAE. To sum up, our model integrates each intermediate feature into SVD++ separately.

Our model can be split into two stages:pre-train stage and re-train stage. These two stage are independent with two loss function (see Eq. (3) and Eq. (7)). In the pre-train stage, we use Denoising AutoEncode to generate ItemFeature $dae(x)$ in hidden layer of DAE, the loss function is Eq. (3). While in the re-train stage, we will update all parameters. Among these parameters, we model the useful features in intermediate features via the attention parameters, the loss function of this step is Eq. (7). By doing this, we can model these intermediate features importance finer-grained and automatically. The prediction score of our method is calculated as:

$$\hat{r_{ui}} = \mu + b_u + b_i + (f_i \odot dae(x_i) + q_i^T) \cdot (p_u + |N(u)|^{-\frac{1}{2}} \cdot \sum_{j \in N(u)} y_j) \tag{6}$$

Where $f_i \in \mathbb{R}^{N \times k}$ represents the attention parameters of intermediate feature of the item entity i, $dae(x_i)$ denotes the intermediate features of item entity i learn from DAE. We use $\odot$ to denote the dot multiplication of matrix. The dot multiplication of f_i and $dae(x_i)$ in the Eq. (6) represents that we assign different weights to each element of the intermediate feature, in this way, we

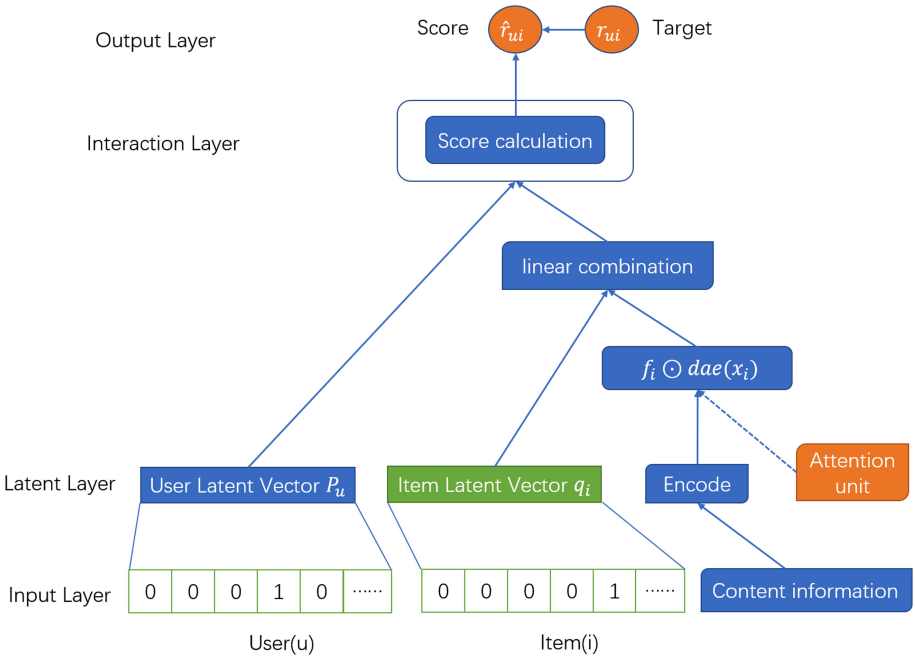

Fig. 2. The architecture of our proposed model HCF-DAE

name f_i as attention unit of intermediate features. We calculate f_i and $dae(x_i)$ in a dot-multiplication manner, so that while improving performance, we also take into account the complexity of our approach. We use stochastic gradient descent (SGD) to update f_i at the element level, and the derivation process is shown in the next section.

3.2 Optimization

Firstly, we use the DAE to obtain the intermediate features of the items the loss function is Eq. (3), and then, we optimize the model in the following way. We learns the parameters by minimizing the regular squared error loss function, which is defined as follows:

$$\underset{b^*,p^*,q^*,y^*}{\arg\min} \sum_{(u,i)\in I} (r_{ui} - \hat{r_{ui}})^2 + \Lambda \cdot f_{reg} \tag{7}$$

The first term of Eq. (7) inherits the error function construction of SVD++. Where Λ represents the weight of regularization parameters, we set it to a single value, f_{reg} is a regularization term of all updated accessories, we set it to avoid model overfitting. f_{reg} expands as follows:

$$f_{reg} = \|f_i\|^2 + b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2 + \sum_{j\in N(u)} \|y_j\|^2 \tag{8}$$

We use SGD to minimize the loss function (see Eq. (7)), and the algorithm iterates over all ratings in the training dataset. First, we denote the prediction error in this way:

$$e_{ui} \stackrel{\text{def}}{=} r_{ui} - \hat{r_{ui}} \tag{9}$$

Then it modifies the parameters by moving in the opposite direction of the gradient, yielding:

$$b_u \leftarrow b_u + \eta_1 \cdot (e_{ui} - \beta_1 \cdot b_u) \tag{10}$$

$$b_i \leftarrow b_i + \eta_1 \cdot (e_{ui} - \beta_1 \cdot b_i) \tag{11}$$

$$q_u \leftarrow q_u + \eta_2 \cdot (e_{ui} \cdot (f_i \odot dae(x_i) + p_i) - \beta_2 \cdot q_u) \tag{12}$$

$$p_i \leftarrow p_i + \eta_2 \cdot (e_{ui} \cdot (q_u + |N(u)|^{-\frac{1}{2}} \cdot \sum_{j \in N(u)} y_j)) \tag{13}$$

From Eq. (10) to Eq. (13), b_u, b_i, q_u, p_i inherits the definition of SVD++. Latent vector update method of user interaction item is shown as follows:

$$\begin{aligned} &\forall j \in N(u): \\ &y_j \leftarrow y_j + \eta_2 \cdot (e_{ui} \cdot |N(u)|^{-\frac{1}{2}} \cdot (f_i \odot dae(x_i) + p_i) - \beta_2 \cdot y_j) \end{aligned} \tag{14}$$

Equation (14) is used to update y_j, where j belongs to user history click item. Similarly, we use the back propagation algorithm to derive the updating method of attention parameters f_i:

$$f_i \leftarrow f_i + \eta_3 \cdot (e_{ui} \cdot (p_i + f_i \odot dae(x_i)) \cdot (q_u + |N(u)|^{-\frac{1}{2}} \cdot \sum_{j \in N(u)} y_j)) \tag{15}$$

From Eq. (10) to Eq. (15), we use η_1, η_2 and η_3 to express the learning rates of parameters updating, β_1 and β_2 are the regularization parameters. We use Eq. (10) to (13) to update parameters b_u, b_i, q_u, p_i. Equation (15) is the process of updating the attention unit f_i of intermediate features. The above formula is updated along the opposite direction of gradient. We will introduce our experiments in the next part.

4 Experiments

When we optimize the parameters, both SGD and GD can be selected. When SGD is used to adjust parameters, one batch of data is used at a time. While GD is used for optimization, all training data are used in each iteration. Due to this, we employ SGD to optimize all parameters. Specific description, for items, we use DAE to option intermediate features. For each user u, we compute the average ratings μ. After that, we begin to update all parameters (see Algorithm 1).

4.1 Experiment Settings

Datasets Description. We evaluate the effectiveness of our proposed model on two public datasets. Movielens is the datasets of ratings to movie, which has been widely utilized to investigate the performance of many recommendation algorithms. This paper uses two stable benchmark Movielens datasets: Movielens-100k and Movielens-1M. Movielens-100k includes 100,000 ratings for 1,660 movies by 943 users, and Movielens-1M contains 1,000,209 ratings of 3,706 movies from 6,040 users. In addition to rating information, Movielens-100k also contains content information for items and users. In this paper, we use genres, year and items ID as content information of items [23]. In the next section, we would define the evaluation metrics of our model.

Algorithm 1. Training algorithm of our model HCF-DAE

procedure UPDATE PARAMETERS
  initial the parameters $q_u, p_i, b_u, b_i, y_i, f_i$.
  generate ItemFeature $dae(x)$.
  for all user u **do**
    for all training samples of user u **do**
      compute sum ratings r_u of user u.
      compute num ratings n_r of user u.
    end for
    compute average ratings of user u:
    $\mu_u = \frac{r_u}{n_r}$.
  end for

  repeat
    for all user u **do**
      for all training samples of user u **do**
        upadate parameters q_u, p_i, b_u, b_i:
        from Equation (10) to Equation (13).
        upadate parameters f_i:
        Equation (15).
      end for
      for all training samples of user u **do**
        upadate parameters y_j:
        Equation (14).
      end for
    end for
  until epoch>=epochs
end procedure

Evaluation Metrics. We use the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) as the evaluation metrics of experiment. Which

has been widely utilized to evaluate the performance of the CF recommendation. RMSE is define as:

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in I}^{M,N}(r_{ui} - \hat{r_{ui}})^2}{|T|}} \tag{16}$$

Where $|T|$ is the number of ratings in the test dataset, and r_{ui} denotes the rating user u to item i. $\hat{r_{ui}}$ represents the corresponding prediction rating. Which represents the error between the predicted value and the true value. The MAE is expressed as:

$$MAE = \frac{\sum_{(u,i)\in I}^{M,N} |r_{ui} - \hat{r_{ui}}|}{|T|} \tag{17}$$

It denotes the average value of the absolute errors of the predicted and observed ratings. The definition of r_{ui} and $\hat{r_{ui}}$ are same as RMSE. Our goal is to reduce the value of RMSE and MAE.

Compared Methods. To verify the efficiency of our model, we compare the proposed model with the following models:

- Funk-SVD [4]. The process of singular value decomposition is simplified into two low-rank matrices, and the two matrices are applied to represent the latent factor of users and items respectively. Then use the two matrix inner products to predict the rating, and obtain the recommending item list.
- SVD++ [9]. This model extends Biased SVD, increases users and item offsets, global average value of rating and historical latent feedback information from users, which has achieved good results.
- AutoSVD++ [23]. This model use AutoEncoder to obtain item intermediate features, then, it integrates the item intermediate features into SVD++, thus modeling the rich content information of the item, and alleviating the cold start problem to a certain extent. Compared with SVD++, Auto-SVD++ has made better experimental results.
- AutoRec [12] utilizes the reconstruction characteristics of the AutoEncoder, proposes two collaborative filtering variants:user-based (U-AutoRec) and item-based (I-AutoRec) that respectively take the partially collected user vector or item vector as input. Our experiments only observe the performance of U-AutoRec.

Parameter Settings. We apply grid search to find the optimal parameters. We tuned the learning rates η_1, η_2, η_3 in the range of [0.001,0.002,...,0.009,0.01], and the value of β_1, β_2 was searched in [0.001,0.002,...,0.01]. For AutoSVD++, we inherit all parameters in [23]. After experiments, the parameters of our model (HCF-DAE) and compared method has be determined and showed in the Table 1.

Table 1. Parameter setting for comparison model and our method

Parameters	η_1	η_2	η_3	β_1	β_2
HCF-DAE	0.005	0.007	0.005	0.005	0.015
AutoSVD++	0.007	0.007	–	0.005	0.015
SVD++	0.005	0.008	–	0.005	0.015
Funk-SVD	–	0.01	–	–	0.01

4.2 Performance Comparison

In this subsection, we summarize the overall experiments of our model and compared models on two movielens test sets respectively. We conduct experiments to check the performance of our model. [26] suggests that sampling should be avoided for metric calculation, nevertheless, sampling is unavoidable in our experiments. In order to ensure the accuracy and credibility of the experimental results, we do five experiments in each group to find the average value of the evaluation metric. Figure 3 shows performance comparison between HCF-DAE and Auto-SVD++ on the Movielens-1M dataset. The experimental results show that AutoSVD++ iterative converges after 15 iterations, and our model iterates 20 times to converge. The final result of HCF-DAE is better than Auto-SVD++ and other two compared models. While the performance of HCF-DAE is not significantly improved compared to SVD++ in Movielens-100k dataset. We speculate that this is due to the small scale of Movielens-100k dataset. This also reflects that AutoEncoder is suitable for model large-scale and complex dataset, while it's difficult to take advantage of deep learning frameworks when AutoEncoder process small-scale datasets. We find that different datasets partition has an effect on the model performance. Figure 4-1 shows experiments on the Movielens-1M dataset. We divide the dataset into test and training dataset in different proportions, and on the premise that the HCF-DAE performs best, the model performs poorly with the training dataset becomes smaller. We analyze that the training dataset is small in size and the model does not have sufficient data for training, so the ideal result cannot be obtained on the test dataset.

4.3 Impact of Attention Unit

We use the Algorithm 1 to iterate the datasets for 30 times, then, we obtain the trajectory of the test error. In order to make the experimental results more reliable, we train five times on the datasets, and we take the average value of RMSE and MAE. Table 2 and Figs. 5, 6 shows the experimental results of our model and the compared methods on these two datasets. We found that the final performance of our approach on both two datasets are the best. Even compared with AutoSVD++, the performance of our approach has also been improved. In the macro point of view, we analyze that comparison with unified weight modeling intermediate features, we use attention unit to model intermediate features

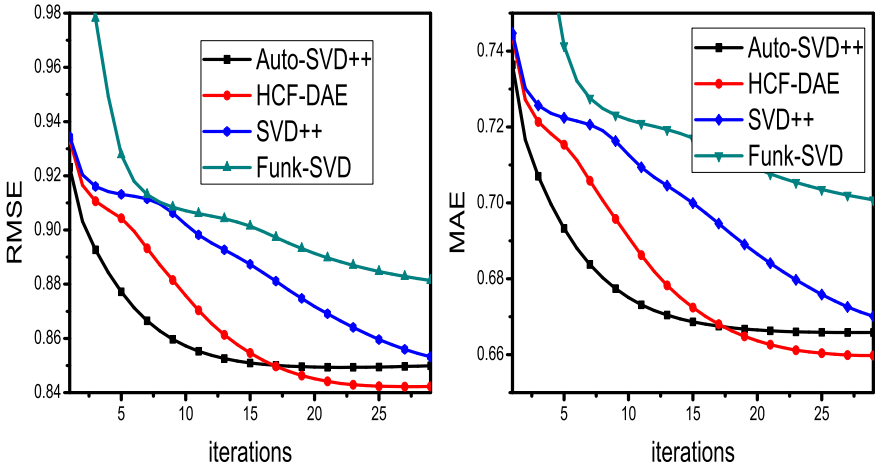

Fig. 3. 1). Test error (RMSE) comparison of each epoch of HCF-DAE and comparison models on Movielens-1M dataset. 2). Performance comparison of each epoch of HCF-DAE and comparison models with the evaluation metrics of average MAE.

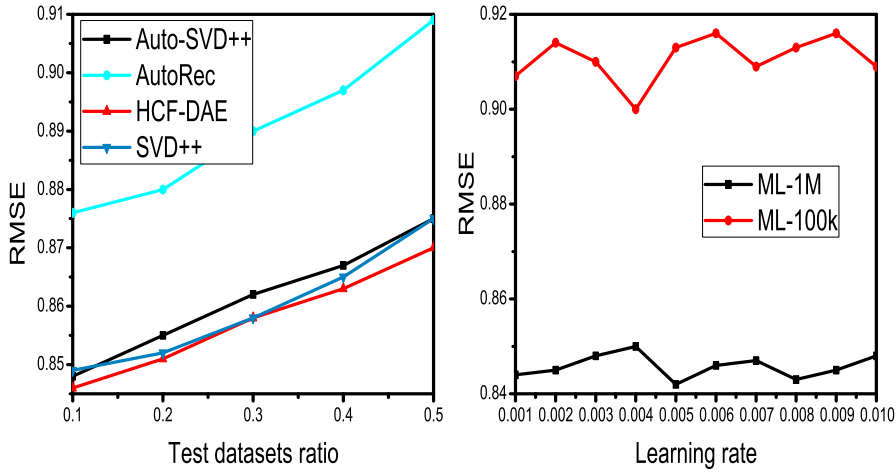

Fig. 4. 1). Average error (RMSE) of each epoch of HCF-DAE and comparison models on Movienens-1M by splitting test dataset with different proportions. 2). Effect of attention parameter learning rate on HCF-DAE performance.

importance is successful. From the micro-level, we analyze that the key advantage of HCF-DAE is ability in interpreting the attention weights of intermediate features. In conclusion, by introducing attention unit, our model can distinguish useful features from noise features, thus the intermediate features obtain from the AutoEncoder can seamlessly integrated with the SVD++ frame work.

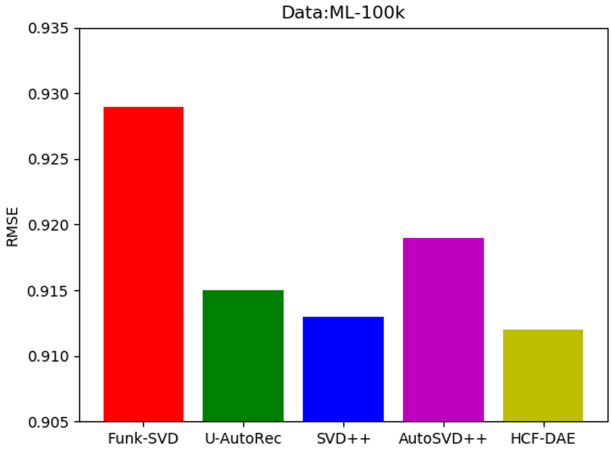

Fig. 5. Histogram of test error of HCF-DAE and comparison models on ML-100k dataset with the evaluation metrics of RMSE.

Table 2. Final performance comparison of our method and compared model for Movielens-1M and Movielens-100k.

	ML-1M		ML-100k	
Model name	RMSE	MAE	RMSE	MAE
Funk-SVD	0.880	0.700	0.929	0.733
U-AutoRec	0.874	0.687	0.915	0.720
SVD++	0.855	0.672	0.913	0.719
AutoSVD++	0.848	0.666	0.919	0.723
HCF-DAE	0.846	0.665	0.912	0.718

4.4 Impact of Different Learning Rates of Attention Unit

We observe the effect of different learning rates on model performance by changing the η_3, which is the attention parameters of the intermediate features. In the Fig. 4-2 we found that when the experiment is performed on ML-1M, the value of $\eta_3 = 0.005$ can let HCF-DAE achieve the best performance. While the experiment on ML-100k, the performance of HCF-DAE is best when $\eta_3 = 0.004$. This shows that the learning rate of attention parameters will fluctuate with the change of dataset. Meanwhile, experiments on ML-1M (see Fig. 4-2) indicated that no matter what value the learning rate of the attention parameters is set, the performance of HCF-DAE is better than SVD++, and in most cases, HCF-DAE performance is better than AutoSVD++. It shows that our model has made a better improvement, and the idea we put forward worked well in the model.

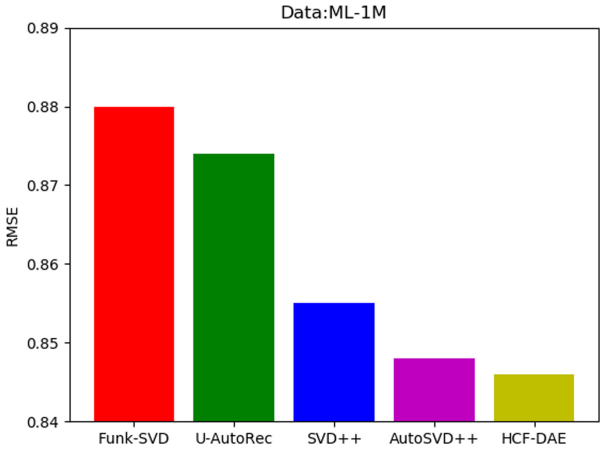

Fig. 6. Histogram of performance of HCF-DAE and comparison model on ML-1M dataset with the evaluation metrics of RMSE.

5 Conclusion

In this paper, we propose a new method to solve the problem of unified weight when AutoEncoder and SVD++ frame work are merged. Our approach uses a clever fusion strategy to learn the intermediate features importance. Instead of manually setting, the attention parameters of the intermediate features in our model are estimated independently, this can more effectively model item feature. We have conducted extensive comparison experiments on two public datasets. The results show that our approach obtains the better prediction results in the comparison model. In future, we plan to explore deep version for our work in the following two directions. First, we intend to further improve the fusion strategy of AutoEncoder and matrix factorization recommendation models. We try to use recurrent neural network to denotes attention strategy. Second, we will take measures to reduce the computational complexity of the algorithm to make model better applied to large-scale datasets.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China under Grant 61862013, 61662015, U1811264, U1711263, and in part by the Natural Science Foundation of Guangxi Province under Grant 2020GXNSFAA159117 and Grant 2018GXNSFAA281199.

References

1. Cao, D., He, X., Miao, L., An, Y., Yang, C., Hong, R.: Attentive group recommendation. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 645–654 (2018)
2. Chen, W., Zhan, L., Ci, Y., Lin, C.: FLEN: leveraging field for scalable CTR prediction. arXiv preprint arXiv:1911.04690 (2019)

3. Cheng, Z., Ding, Y., He, X., Zhu, L., Song, X., Kankanhalli, M.S.: Aˆ3NCF: an adaptive aspect attention model for rating prediction. In: IJCAI, pp. 3748–3754 (2018)
4. Funk, S.: Netflix update: try this at home. https://sifter.org/~simon/journal/20061211.html. Accessed 11 Dec 2006
5. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182 (2017)
6. He, X., Tang, J., Du, X., Hong, R., Ren, T., Chua, T.S.: Fast matrix factorization with nonuniform weights on missing data. IEEE Trans. Neural Netw. Learn. Syst. **31**, 2791–2804 (2019)
7. Huang, T., Zhang, Z., Zhang, J.: FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In: Proceedings of the 13th ACM Conference on Recommender Systems, pp. 169–177 (2019)
8. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery And Data Mining, pp. 426–434 (2008)
9. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
10. Liu, B., et al.: AutoFIS: automatic feature interaction selection in factorization models for click-through rate prediction. arXiv preprint arXiv:2003.11235 (2020)
11. Pan, J., et al.: Field-weighted factorization machines for click-through rate prediction in display advertising. In: Proceedings of the 2018 World Wide Web Conference, pp. 1349–1357 (2018)
12. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: AutoRec: autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web, pp. 111–112 (2015)
13. Song, W., Shi, C., Xiao, Z., Duan, Z., Xu, Y., Zhang, M., Tang, J.: AutoInt: automatic feature interaction learning via self-attentive neural networks. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 1161–1170 (2019)
14. Strub, F., Gaudel, R., Mary, J.: Hybrid recommender system based on autoencoders. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp. 11–16 (2016)
15. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, 5–9 June 2008 (2008)
16. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)
17. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S.: KGAT: knowledge graph attention network for recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 950–958 (2019)
18. 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, pp. 153–162 (2016)
19. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.S.: Attentional factorization machines: learning the weight of feature interactions via attention networks. arXiv preprint arXiv:1708.04617 (2017)

20. Xin, X., He, X., Zhang, Y., Zhang, Y., Jose, J.: Relational collaborative filtering: modeling multiple item relations for recommendation. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 125–134 (2019)
21. Xue, F., He, X., Wang, X., Xu, J., Liu, K., Hong, R.: Deep item-based collaborative filtering for top-n recommendation. ACM Trans. Inf. Syst. (TOIS) **37**(3), 1–25 (2019)
22. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: a survey and new perspectives. ACM Comput. Surv. (CSUR) **52**(1), 1–38 (2019)
23. Zhang, S., Yao, L., Xu, X.: 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, pp. 957–960 (2017)
24. Zhang, S., Yao, L., Xu, X., Wang, S., Zhu, L.: Hybrid collaborative recommendation via semi-autoencoder. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.S. (eds.) Neural Information Processing, pp. 185–193. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70087-8_20
25. Song, Q., Cheng, D., Zhou, H., Yang, J., Tian, Y., Hu, X.: Towards automated neural interaction discovery for click-through rate prediction. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 945–955 (2020)
26. Krichene, W., Rendle, S.: On sampled metrics for item recommendation. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1748–1757 (2020)
27. Ji, S., Feng, Y., Ji, R., Zhao, X., Tang, W., Gao, Y.: Dual channel hypergraph collaborative filtering. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Ser. KDD 2020, pp. 2020–2029 (2020)
28. Hidasi, B., Karatzoglou, A.: Recurrent neural networks with top-k gains for session-based recommendations, pp. 843–852 (2017)