ITC 4/48
Information Technology and Control
Vol. 48 / No. 4 / 2019
pp. 682-693
DOI 10.5755/j01.itc.48.4.23149

AANMF: Attribute-Aware Attentional Neural Matrix Factorization

Received 2019/04/10 | Accepted after revision 2019/11/19

http://dx.doi.org/10.5755/j01.itc.48.4.23149

AANMF: Attribute-Aware Attentional Neural Matrix Factorization

Bo Zheng
South China University of Technology; e-mail: holy5pb@163.com

Jinsong Hu
South China University of Technology; e-mail: cshjs@scut.edu.cn

Corresponding author: cshjs@scut.edu.cn

Matrix Factorization (MF) is one of the most intuitive and effective methods in the Recommendation System domain. It projects sparse (user, item) interactions into dense feature products which endues strong generality to the MF model. To leverage this interaction, recent works use auxiliary information of users and items. Despite effectiveness, irrationality still exists among these methods, since almost all of them simply add the feature of auxiliary information in dense latent space to the feature of the user or item. In this work, we propose a novel model named AANMF, short for *Attribute-aware Attentional Neural Matrix Factorization*. AANMF combines two main parts, namely, neural-network-based factorization architecture for modeling inner product and attention-mechanism-based attribute processing cell for attribute handling. Extensive experiments on two real-world data sets demonstrate the robust and stronger performance of our model. Notably, we show that our model can deal with the attributes of user or item more reasonably. Our implementation of AANMF is publicly available at https://github.com/Holy-Shine/AANMF.

KEYWORDS: Recommender system, Matrix factorization, Neural network, Attention mechanism, Collaborative filtering.

1. Introduction

The critical problem of the recommender system is how to model the interactions of user and item—a high dimensional sparse interaction matrix. To address this computational complexity problem, Koren et al. proposed the matrix factorization method [16]. It projects users and items into a dense latent space. Thus a user or an item can be represented by a dense feature vector, and the inner product by two features from a user and an item can show the degree of his/her favorability towards this given item. To improve the performance of the model, several works explored the advantage of explicit social information such as a user's relationship to improve recommendation accuracy [3, 8, 27, 34]. Some works also tried to utilize attribute information so that latent features could have stronger expressive ability [15, 17, 31, 32, 35]. SVD++ [15] is a classical method which well models auxiliary information i.e. attributes of the user or item.

Although SVD++ performs better than other models that do not use attributes, we argue that irrationality still exists in its direct addition way. To illustrate it, we take attributes of users as an example. According to SVD++-like attribute-aware models, a specific user's attributes will get the same latent feature vector no matter what the item is. That is, towards each item, a certain user's attributes will provide the same bias, which does not reflect the reality. Let us consider a realistic scene of predicting people's preference for movies. We choose gender as that specific user attribute. If the recommender system predicts people's rating to a Marvel's movie, the audience's gender usually is not a factor affecting preference, whereas when it comes to a romantic love movie, the system should pay more attention to the user's gender.

To make our model able to catch this information, we introduce the attention mechanism. Attention mechanism is a widely used method in NLP and Compute Vision domain [1, 26, 30, 33]. For example, Yin et al. utilized the attention mechanism to model language sequence [29], and Xu et al. used it to generate image caption [26]. Some recent works also introduce attention mechanism into their matrix factorization model. Chen et al. [4] and Xiao et al. [25] combined neural networks and attention mechanism, hence improved the performance of their model. However, these works concentrated on the final latent feature of user and item, with regards to modeling the attributes, they still used shared weight embedding or do nothing.

In this work, we propose a novel model named AANMF. It combines two main parts, namely, the neural-network-based factorization machine and the attention-mechanism-based attributes processing unit. Extensive experiments on two real-world data sets demonstrate robust and stronger performance of our model. Notably, we show that our model can deal with user or item's attributes more reasonably.

2. Related Work

With the success of the attention mechanism in the field of image [26] and NLP [29], more and more works began to introduce it into other fields. In the recommender system domain, two works applied this technology to model and enhanced the performance of their proposed FM framework. We simply summarize their works next and clarify the difference between our model and them.

Attentive Collaborate Filtering (ACF) [4], proposed by Chen et al., introduced attention mechanism (we called it AM in the following paper) into the recommender system with multimedia content problem. They utilized AM to filter some useless implicit information. With regard to explicit information and (attribute, user/item) interaction modeling, their work does not involve them.

Another attention-based work is called AFM [25], which was proposed by Xiao et al. Their work focused on the interaction between user and item in the latent feature space. As for how to embed origin categorical variables into latent space, they still used shared weight.

As these two attention-net-based methods are probabilistic models and predict an item ranking list rather than a rating prediction, we do not choose them as baselines in this work.

Some works were proposed to enhance the performance of the MF model. Del Corso et al. [6] built an NMF model that change adaptively the function to be minimized at each step. Wang et al. [23] proposed an MF model named LOD-MF, which dug out implicit feedback information and applied a hybrid similarity measure to identify the semantically similar neighbors of the target item. Cai et al. [2] introduced a novel parameter tuning framework named mrMoulder. It can recommend an optimized configuration for a new job in a short time. All of their works focus on the training stage while our method focuses on the model building stage.

In addition to these, some works also expanded attribute modeling to improve prediction accuracy. Some recommendation models are based on ratings, reviews, and social relationships [13, 24]. These relations can be treated as another user/item attribute entry in our model. Yang et al. [28] introduced the cloud platform into the multimedia platform which explored the benefit of mobile cloud computing.

3. SVD++

SVD++ is an attribute-aware model that can handle the attributes of both sides of the interaction. Traditional MF methods only regard the product of latent space features from user and item as the model's prediction outcome. SVD++ assumes that the attributes provide bias to the final prediction. For each user's attribute, SVD++ projects it into latent space and gets a dense feature, then SVD++ models user feature by adding these features to the user's original feature:

$$\hat{y}_{ui} = \underbrace{\mu + b_i + b_u}_{\text{bias}} + \underbrace{\mathbf{q}_i^T \left(\mathbf{p}_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} \mathbf{a}_j \right)}_{\text{inner production}}, \tag{1}$$

where μ is the global bias, b_i and b_u denote item bias and user bias, respectively. As for the key part, assuming that SVD++ describe a K-dim latent space $\mathbb{R}^K$, correspondly K represents K latent factors. $\mathbf{q}_i \in \mathbb{R}^K$ and $\mathbf{p}_u \in \mathbb{R}^K$ denote item and user's original dense feature, respectively. I_u is the set of user attributes, $\mathbf{a}_j \in \mathbb{R}^K$ is the i-th latent feature of the attribute. Almost all of MF methods are based on this equation.

It is worth noting that this model ignores the correlation between item feature and user attributes i.e., no matter what the item is, the user's attributes will contribute the same bias to the final prediction—which is unrealistic and makes the model not being able to take full advantage of attribute information. For a movie recommendation example, the age of audience should contribute little bias to action movies, whereas there should be big bias to romantic movies. However, models like SVD++ share the same weight for attributes. That is, latent features among set are neither dynamical nor configurable, which limits the generalization performance of the model.

Let us discuss the reason why non-configurable I_u limits the performance of the model with another perspective. By simplifying the SVD++ model and analyzing its nature, we can rewrite Equation (1) as follows:

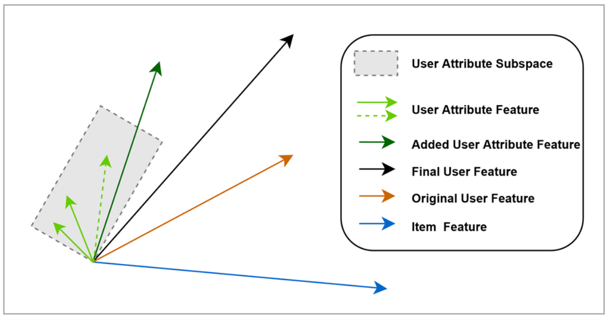
$$\hat{y}_{ui} = \mathbf{bias} + \left\langle \mathbf{q}_i, \mathbf{p}_u + \sum_{k=1}^{|I_u|} \lambda_k \mathbf{a}_k \right\rangle, \tag{2}$$

where $\langle \cdot, \cdot \rangle$ is the dot product of two vectors; λ_k is the normalization parameter. This equation shows that the final prediction actually is the distance between the feature vectors of a user and an item.

Consider a low dimension case. Figure 1 illustrates the primary process of this model in a low dimension feature space. The gray dash box denotes attribute subspace, and the dark green arrow indicates the sum of features of all attributes. After simply adding $\sum \lambda \mathbf{a}$ and user feature $\mathbf{p}_u$, the model gets a new vector $\mathbf{p}'_u$. Then the distance between item feature $\mathbf{q}_i$ and $\mathbf{p}'_u$ can be seen as the final prediction. Figure 1 also shows that SVD++ assumes that each attribute contributes the same bias to the final prediction (same normalization parameter λ).

Figure 1

The example illustrates SVD++'s limitations. Since all attribute features are simply added in feature space, the aggregate final feature will fluctuate



However, when some attribute is not the main impact factor, this model will get an unreasonable result. Assume the dashed arrow in the box is the "some attribute"—user occupation. No matter what direction the dash arrow points (different occupations get different feature vectors), we expect the model will get the same prediction. As λ is not configurable, $\mathbf{p}'_u$ will be fluctuant, which causes the model to get a siginificant loss of prediction. Hence, we proposed a model which uses the attention mechanism to handle this vital parameter λ.

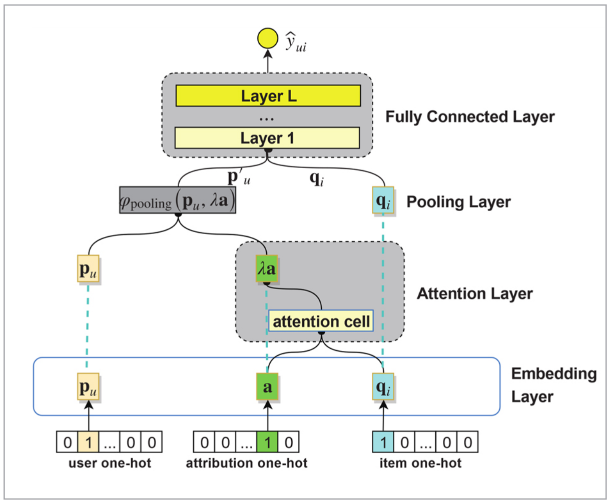
4. Attributes-Aware Attentional Neural Matrix Factorization

Recently, several works have applied neural networks to matrix factorization. Neural networks have been proven to be capable of approximating any continuous function [12]. In addition, they can also generalize better to unseen feature combinations through low dimensional dense embeddings learned for the sparse features [5]. Hence, we divide our model into two main parts: the first part for generalized matrix factorization and the other for handling attention-based attribute modeling.

4.1. Model

Figure 2 illustrates our proposed AANFM model. For clarity purposes, we omit the item attribute input and only show the calculation process of one attribute in Figure 2. On the whole, our AANMF model embeds sparse one-hot representation of user and item to dense vectors. Then it merges these two vectors and gets the final prediction. We call it *Neural network based Factorization Machine* (NFM), which is inspired by He et al. proposed the NCF model [10]. In their work, NFM was proved that could mimic a large family of factorization models. On this basis, we introduce the attention mechanism into our model for decreasing the irrationality of MF models, which is the main contribution of this paper.

Figure 2
The AANMF framework



Additionally, inspired by [24], several pooling ways have been applied so that the model can get better performance. In the following part of this section, we detail our model from bottom to top.

Input and Embedding Layer

There is an enormous drawback to one-hot representation, hence, like other MF models, we maintain an embeds table where each row represents a dense feature, which is similar to the distributed representation of words [18]. In our model, we use the Embedding layer to implement this requirement and regard one-hot input as a retrieval index. The corresponding row in the learnable embeds table can be seen as the latent vector for a user or an item in the context of the latent factor model.

Attention Layer

Owing to its biological interpretability, attention mechanism has been applied to many neural-network-based tasks [4, 29]. The idea of setting an attention layer (Attention Cell in Figure 2) is to investigate the interaction between user/item and item/user's attributes. That is, the model should learn the weight of the attribute towards different user/item rather than all of the attributes share the same weight.

Algorithm 1 and Figure 2 show the process of attention cell in the AANMF framework. At each attention step, it generates a probability vector λ via learning (item, user attributes) interaction, and this probability vector represents how much this attribute contributes to the user preference latent vector. Hence, according to notational conventions of Equation (1), user's final latent feature can be described by:

$$\mathbf{p}'_u = \varphi_{\substack{\text{pooling} \\ j\in|I_u|}}\left(\mathbf{p}'_u, \lambda_j \odot \mathbf{a}_j\right), \quad (3)$$

Algorithm 1: The process of Attention Cell in AANMF framework.

Input: The set of user's attributes, I_u; Current item's embedding vector, $\mathbf{q}_i$; Latent space dimension, K
Output: Probability vectors represent the model's attention to each attributes, λ;

```
1  For each user attribute $\mathbf{a}_j \in I_u, j \in |I_u|$, concatenate
   $\mathbf{a}_j$ and $\mathbf{q}_i$ into a dense vector: $\mathbf{V}_j = [\mathbf{a}_j, \mathbf{q}_i]$;
2  Use a dense layer with tanh project $\mathbf{V}_j$ into K-dims
   vector $\mathbf{v}_j = \tanh(W_1\mathbf{V}_j + b_1)$;
3  Add a single neural unit dense to each output from
   the last procedure after which applying a softmax
   layer to get attention parameter $\lambda$, whose each entry is equal to:
   $\lambda_j = \frac{\exp(\mathrm{ReLu}(W_2\mathbf{v}_j + b_2))}{\sum_{k=1}^{|I_u|}\exp(\mathrm{ReLu}(W_2\mathbf{v}_k + b_2))}$;
4  Return $\lambda$
```

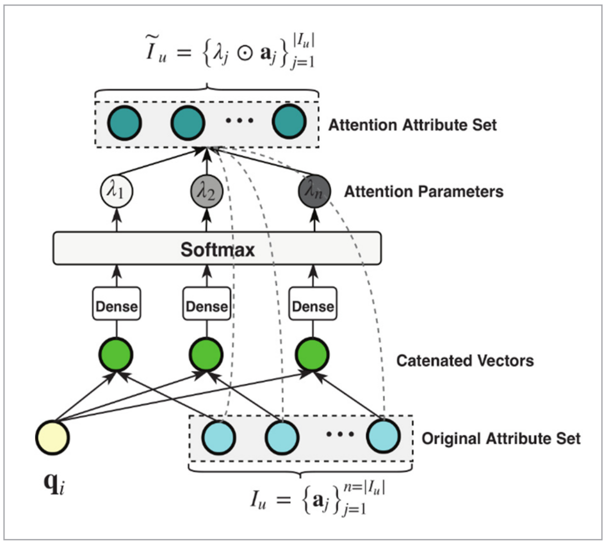
where $\odot$ denotes the element-wise product of two vectors. It is worth mentioning that in Equation (3), we use a well-chosen pooling way to solve two vectors, we left its implementation detail in the next section.

Pooling Layer

Our model will get a set of dense vector pairs $(\mathbf{p}_u, \mathbf{q}_i)$ to describe user u and item i after the embedding layer. Meanwhile, the embedding layer outputs an attribute latent vector set, size of which may vary for different inputs since different user/item may have a different number of attributes.

Figure 3

The process of the Attention Cell



Traditional MF methods simply add (sum-pooling) or average (average-pooling) these vectors into one vector. We argue that it can limit the performance of the model as it ignores the correlation between two vectors in latent space. Inspired by [24], we adopt a pooling way named pair-wise pooling. As illustrated in Figure 3, we let $\tilde{I}_u = \{\lambda_j \odot \mathbf{a}_j\}_{j=1}^{|I_u|}$ denote the user's attributes feature set after attention cell. What pair-wise-pooling does is as follows:

$$\mathbf{p}'_u = \varphi_{\text{pairwise}}(\mathbf{p}_u, \tilde{I}_u) = \sum_{j=1}^{|\tilde{I}_u|} \mathbf{p}_u \odot (\lambda_j \odot \mathbf{a}_j) + \sum_{j=1}^{|\tilde{I}_u|} \sum_{j'=j+1}^{|\tilde{I}_u|} (\lambda_j \odot \mathbf{a}_j) \odot (\lambda_{j'} \odot \mathbf{a}_{j'}). \quad (4)$$

As Equation (4) shows, pair-wise pooling can encode correlation between them in the feature latent space.

Fully Connected Layer

In order to endow AANMF modeling with a high level of non-linearities, a stack of fully connected layers is added above the pooling layer. Firstly, model merges 2 vectors: $\mathbf{p}'_u$, $\mathbf{q}'_i$ with the element-wise product (*cf.* Figure 2), then make result vector flows through L fully connected layers and get the final prediction $\hat{y}_{ui}$. Each layer has weight matrix W_l, bias b_l, activation function σ_l, where l denotes the l-th layer:

$$\hat{y}_{ui} = \sigma_L(\cdots \sigma_1(W_1(\mathbf{p}'_u \odot \mathbf{q}_i) + b_1)\cdots). \quad (5)$$

We choose ReLu as the non-linear unit since ReLu can avoid vanishing gradient and exploding gradient problem [7].

Algorithm 2: The process of learning

Input: User feature matrix $\mathbf{P} = \{\mathbf{p}_j\}_{j=1}^{n_u}$, Item feature matrix $\mathbf{Q} = \{\mathbf{q}_j\}_{j=1}^{n_i}$, Neural layer weights $\mathbf{W} = \{W_{t_1}, W_{t_2}, W_{l_1}, W_{l_2}, ..., W_{l_L}\}$, bias $\mathbf{b} = \{b_{t_1}, b_{t_2}, b_{l_1}, b_{l_2}, ..., b_{l_L}\}$, learning rate α

Output: Updated $\mathbf{P}, \mathbf{Q}$

```
1  Initailize P, Q, W, b
2  for $j = 1; j <$ iterations; $j$++ do
3      for each user, item pair $(u, i)$ do
4          Get their latent space features $\mathbf{p}_u$ and $\mathbf{q}_i$ from P, Q;
5          Compute $\lambda$ using Algorithm 1;
6          Use pair-wise pooling(Equation (6)) to compute the final concatenated vector;
7          Compute prediction, loss and add each loss to total loss $L$
8      end for
9      for $W$ in W, $b$ in b do
10         $W \leftarrow W - \alpha \frac{\partial L}{\partial W}, b \leftarrow b - \alpha \frac{\partial L}{\partial b}$
11     end for
12     for p in P, q in Q do
13         $\mathbf{p} \leftarrow \mathbf{p} - \alpha \frac{\partial L}{\partial \mathbf{p}}, \mathbf{q} \leftarrow \mathbf{q} - \alpha \frac{\partial L}{\partial \mathbf{q}}$
14     end for
15 end for
16 Return P, Q
```

4.2. Learning

We divide the learning process of our model into three parts: Input, Pooling, and Output.

Input. Most neural network frameworks provide Embedding Layer. It maintains a learnable parameter matrix, each row of which can be regarded as a latent feature. Thus, the input of AANMF can be the one-hot representation of user/item. For example, (user, item) pair (u_j, i_k) participate in the calculation, correspondingly, row j in user's latent feature matrix and row k in item's latent feature matrix would update after one iteration training step.

Pooling. According to Equation (4), we need a nested for-loop in code to accomplish pairwise-pooling. In practice, we adopt another way proposed by [24]:

$$2\mathbf{p}_u^{'} = \left(\mathbf{p}_u + \sum_{j=1}^{|\tilde{I}_u|}\tilde{\mathbf{a}}_j\right) \odot \left(\mathbf{p}_u + \sum_{j=1}^{|\tilde{I}_u|}\tilde{\mathbf{a}}_j\right) - \mathbf{p}_u \odot \mathbf{p}_u - \sum_{j=1}^{|\tilde{I}_u|}\tilde{\mathbf{a}}_j \odot \tilde{\mathbf{a}}_j, \quad (6)$$

where $\tilde{\mathbf{a}}_j = \lambda_j \odot \mathbf{a}_j$. It is worth noting that Equation (6) can be computed in $O(K\,|\,\tilde{I}_u\,|)$ time.

Output. Unlike some ranking models [22], BPR [21] and eALS [11], we train AANMF model to accomplish a rating task; that is, given a user and an item, the trained model will get a user's rating score prediction towards to item. Hence, we opt for MSE as loss function:

$$\mathcal{L}(\hat{y}, y) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \quad (7)$$

Additionally, a computationally efficient gradient descent method—Adam [14] is adopted as the optimizer to speed up the convergence of our model. We conclude the learning process in Algorithm 2.

5. Experiment

5.1. Data Set

Several data sets are used to evaluate the performance of recommendation methods, but only MovieLens100K and MovieLens1M contain user-attribute information. Hence, we evaluate the performance of AANMF on these two real-world data sets. The basic information on these two data sets is provided in Table 1.

Table 1
Statistics of the dataset

Name	User#	Item#	Interaction#
MovieLens100K	943	1682	100000
MovieLens1M	6040	3952	1000309

MovieLens100K and MovieLen1M are two subsets of data obtained from the MovieLens[1] research project. They contain 100 thousand and 1 million rating records with format (*u*, *i*, *rating*), respectively. As the attributes of movies in Movielens are few, we opt for the user's attributes set as one input channel of the AANMF model. For convenience, three attributes of the user are chosen. We illustrate their statistics in Table 2.

Table 2
Statistics of the attributes

User Attribute	#	Class	Encoding
User gender	2	Male	01
		Female	10
User age	7	Under 18	0000001
		18-24	0000010
		25-34	0000100
		...	...
User job	21	Artist	$\underbrace{00\cdots01}_{21}$
		Lawyer	$\underbrace{00\cdots10}_{21}$
		...	...

User gender:2 represents each user's gender, and has two possibilities, male or female. While 21 represents the number of classes of jobs, which is 21 (doctor, artist, *etc.*). For user age, we divide the age range from 0 to over 56 into 7 age groups (under 18, 18-24, 25-34, *etc.*) In order to adapt the data to the input of the model, we reorganize the dataset shown in Table 3.

1 http://www.grouplens.org

5.2. Baseline

As the contribution of this paper is proposing a more reasonable method to model the input of attributes, we opt for some representative models with attribute processing unit as baselines:

- **SVD++**. This is a classic model which well utilizes attributes to enhance its performance.
- **NFM.** Neural Factorization Machine [9] is a neural network-based FM [20] method that has a strong power to model categorical variables, such as the attributes that we mentioned above.
- **AESR.** Autoencoder-based social recommender system [19] is a state-of-the-art method which is a hybrid method that by modeling a joint optimization function extends deep Autoencoder with top-k semantic social information.
- **ACMF.** ACMF [32] is an attribute coupling- based matrix factorization method which incorporates item-attribute information into the matrix factorization model as well as adopts coupled object similarity to capture the relationship among items.

We conduct experiments to verify whether AANMF does better in rating prediction task and answer the following questions:

RQ1: Does AANMF perform better than the traditional attribute-aware models?

RQ2: Does AANMF outperform other state-of-the-art methods?

RQ3: Do the results of our experiments have a strong explanatory power?

5.3. RQ1: Does AANMF Perform Better than the Traditional Attribute-Aware Models?

We choose SVD++ and NFM as typical traditional attribute-aware models. Figure 4 shows the training and testing process of our model on two datasets. We can intuitively observe that the loss curve of AANMF is under the other two curves, which means AANMF can better fit the data and lead to more reliable predictions. Furthermore, although the neural-based model (AANMF&NFM) achieves a higher testing error in the early iteration (seen in subgraph 2 in Figure 4), they get a continuous error decrease in subsequent iterations. We owe a good deal to the generality of the neural network.

We also tune models and report the best performance. Figure 5 shows the best performance of MSE with respect to the number of latent factors on the two datasets mentioned above. In order to facilitate the comparison, NFM and AANMF get the same two fully connected layers before the final output. For SVD++, we simply dot two latent space features into prediction rank.

First, we can see that our AANMF gets the best performance in both two subgraphs. Second, nearly all the curves in Figure 5 have the same trend: for small latent factors, the model will get better performance along with increasing the number of factors, whereas for large latent factors, it begins to overfit. Lastly, a horizontal comparison between two subgraphs shows that the neural-network-based model can model larger datasets easier (on average, the improvement over SVD++ in ml100k and ml1m is 4.5% and 6.5%, respectively).

Table 3
Reorganized Data set

Index	User id	Movie id	Rating	User gender	User age	User job
0	1	1193	5	0	0	10
1	2	1193	5	1	5	16
2	3	1193	4	1	6	12
...	...	...	...	...	...	...
100206	5780	2845	1	1	4	17
100207	5851	3607	5	0	4	20
100208	5938	2909	4	1	6	1

Figure 4

Training and test error *w.r.t.* iterations

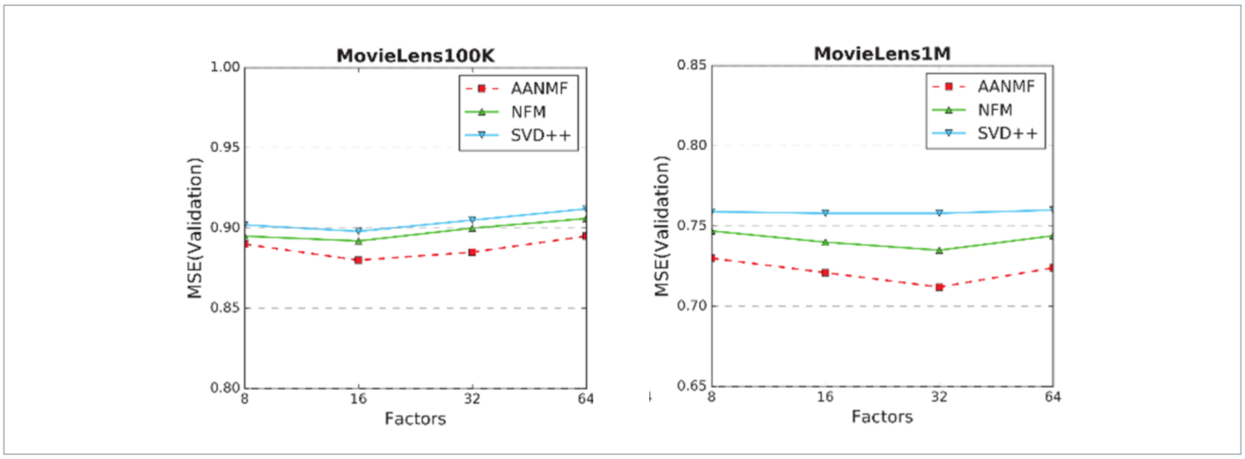
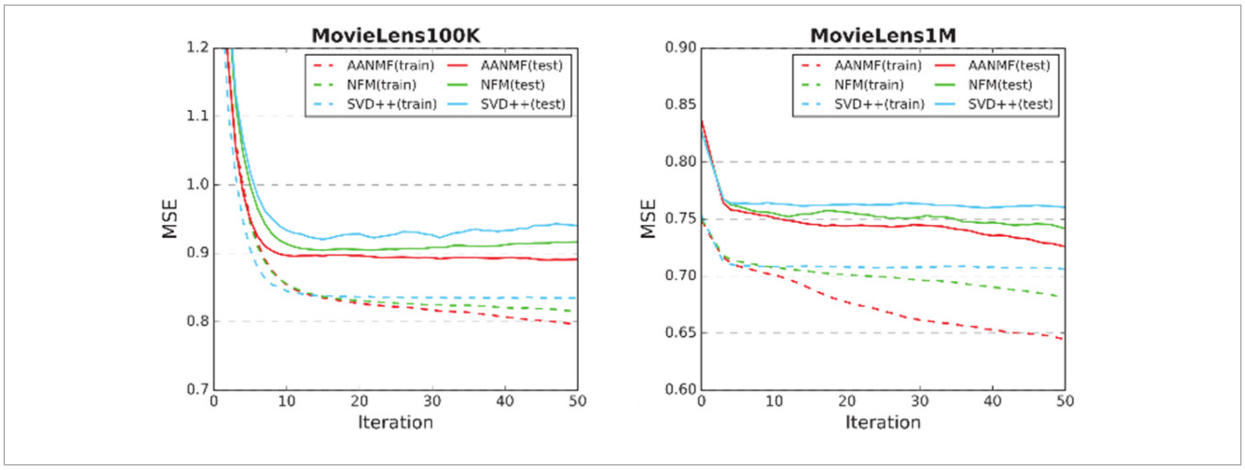


Figure 5

The performance of MSE w.r.t. the number of latent factors



5.4. RQ2: Does AANMF Perform Better than Other State-of-the-Art Methods?

ASER and ACMF are two recently published state-of-the-art algorithms of the recommender system. We choose two popular matrices, namely, *Mean Absolute Error(MAE)* and *Root Mean Squared Error(RMSE)* to measure the recommendation quality:

$$MAE = \frac{\sum_{t=1}^{T} |\hat{y}_t - y_t|}{T}$$
$$RMSE = \sqrt{\frac{\sum_{t=1}^{T} |\hat{y}_t - y_t|^2}{T}}, \qquad (8)$$

where T denotes the number of interaction pairs.

We tune these models to get the best performance on MovieLens data sets and report comparison results in Table 4. It is easy to observe that our AANMF model is superior to the other compared methods. AANMF improves MAE and RMSE by 2.7% and 1.8% on MovieLens100K as well as 5.7% and 3.5% on MovieLens1M. We can also see that pairwise pooling way indeed improves the performance of the model (*cf.* AAANMF+P in Tabel 4).

We use the Nemenyi test to confirm that our model exactly performs better than the methods listed above. We choose as the confidence of the null hypothesis:

Table 4
A comparision of AANMF with the other two methods on MAE and RMSE

Data sets	Metric	ACMF	AESR	AANMF	AANMF+P
MovieLens100K	MAE	0.7282	0.7216	0.7022	**0.7012**
	RMSE	0.9186	0.9176	0.9012	**0.8979**
MovieLens1M	MAE	0.7045	0.6980	0.6580	**0.6473**
	RMSE	0.8875	0.8792	0.8485	**0.8322**

all methods have the same performance. Then after applying the Friedman test, we reject this hypothesis, after which we use the Nemenyi test as a post-hoc test to estimate each method pair. Nemenyi test computes the critical distance CD as follows:

$$CD = q_{\alpha}\sqrt{\frac{k(k+1)}{6N}}, \tag{9}$$

where k is the number of methods and N is the number of data sets. If the distance between the average rank of two methods is below this value, it can be claimed that these two methods have a notable gap.

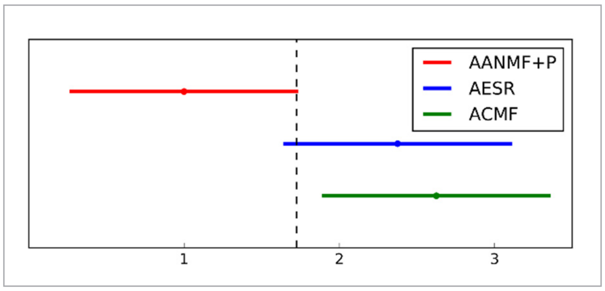
We use 0.01 as threshold to rank these three methods (ACMF, AESR and AANMF+P). That is, if the gab between the value of two methods in Table 4 is lower than 0.01, we treat them as in the same rank. We compute average ranks of these three methods in Table 5.

Table 5
Rank of 3 methods in 2 metrics

Metric	ACMF	AESR	AANMF+P
ML100K-MAE	2.5	2.5	1
ML100K-RMSE	3	2	1
ML1M-MAE	2.5	2.5	1
ML1M-RMSE	2.5	2.5	1
Average rank	2.625	2.375	1

Finally, we compute CD and display the Friedman test result in Figure 6. According to Figure 6, our AANMF+P exactly performs better than the other two methods.

Figure 6
Friedman test result



5.5. RQ3: Do the Results of Our Experiments Have a Strong Explanatory Power?

It is known that a useful property of the attention mechanism is its highly interpretable outputs. To show whether the learned attentions are meaningful, we choose attention weight learned from MovieLen100K dataset as an example.

For the convenience of visualization and observation, we assume that the distance between movies of the same type is short in feature space. We select 8 typical movie types, and for each type, we make movie clusters to train the AANMF model, observe the attention outputs with respect to 3 user attributes, namely, gender, age, and job. Figure 7 shows the average levels among different types of movies (colors white and black represent high attention and low attention, respectively). According to this figure, we can simply find that:

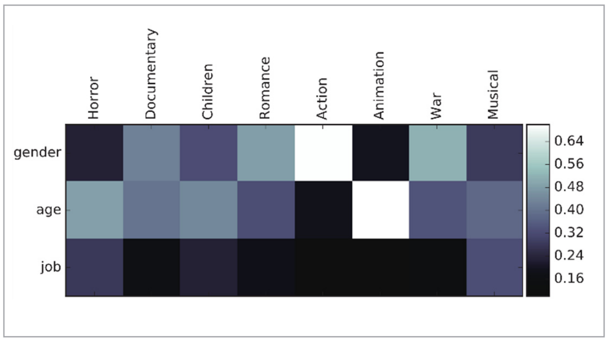
1 Action movies tend to get high attention towards gender, while animation movies get high attention

towards age, which is in line with reality. As for the other types, we can also find reasonable bias with different attributes

2 User job is always not the main impact factor to the final prediction (*cf.* Figure 7, the third row always gets black color). We owe this phenomenon to the fact that someone's job usually does not contribute to his/her rank toward a movie.

Figure 7

Attention weight for different attributes



6. Conclusion and Future Work

In this work, we propose a neural-network-based framework named AANMF, which develops a novel method combining AM and matrix factorization. Extensive experiments on AANMF and baselines demonstrate the rationality and advancement of our model in attribute modeling. We do not focus on tuning the other part of the model as this work puts emphasis on proving the new architecture we proposed actually works.

In the future, we will continue to study how AM can investigate the interaction between user/item and attribute deeper. In addition, as this work serves as a guideline for a combination of AM and NFM, we will attempt to adjust other structures for further study. For example, after the pairwise layer in AANMF, our model simply adds some dense layer onto the element-wise product of user and item. There may exist more reasonable ways of handling this. Moreover, we are interested in exploring how to apply AM in modeling the interaction between user/item and multimedia auxiliary information rather than only categorical attributes.

References

1. Bahdanau, D., Cho, K., Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. Compute Science, 2014.
2. Cai, L., Qi, Y., Wei, W., Wu, J., Li, J. mrMoulder: A Recommendation-based Adaptive Parameter Tuning Approach for Big Data Processing Platform. Future Generation Computer Systems, 2019, 93, 570-582. https://doi.org/10.1016/j.future.2018.05.080
3. Chaney, A. J., Blei, D. M., Eliassi-Rad, T. A Probabilistic Model for Using Social Networks in Personalized Item Recommendation. The 9th ACM Conference. ACM, 2015, 43-50. https://doi.org/10.1145/2792838.2800193
4. Chen, J., Zhang, H., He, X., Nie, L., Liu, W., Chua, T. S. Attentive Collaborative Filtering: Multimedia Recommendation with Item and Component-level attention. Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, 335-344. https://doi.org/10.1145/3077136.3080797
5. Cheng, H. T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Lspir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., Shah, H. Wide & Deep Learning for Recommender Systems. Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, 2016, 7-10. https://doi.org/10.1145/2988450.2988454
6. Del Corso, G. M., Romani, F. Adaptive Nonnegative Matrix Factorization and Measure Comparisons for Recommender Systems. Applied Mathematics and Computation, 2019, 354, 164-179. https://doi.org/10.1016/j.amc.2019.01.047
7. Glorot, X., Bordes, A., Bengio, Y. Deep Sparse Rectifier Neural Networks. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, 15, 315-323.
8. Guo, G., Zhang, J., Yorke-Smith, N. Trustsvd: Collaborative Filtering with Both the Explicit and Implicit Influence of User Trust and of Item Ratings. Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, 123-129.
9. He, X., Chua, T. S. Neural Factorization Machines for Sparse Predictive Analytics. Proceedings of the 40th

International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, 355-364. https://doi.org/10.1145/3077136.3080777

10. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T. S. Neural Collaborative Filtering. Proceedings of the 26th International Conference on World Wide Web, 2017, 173-182. https://doi.org/10.1145/3038912.3052569
11. He, X., Zhang, H., Kan, M. Y., Chua, T. S. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2016, 549-558. https://doi.org/10.1145/2911451.2911489
12. Hornik, K., Stinchcombe, M., White, H. Multilayer Feedforward Networks are Universal Approximators. Neural Networks, 1989, 2(5), 359-366. https://doi.org/10.1016/0893-6080(89)90020-8
13. Ji, Z., Pi, H., Wei, W., Xiong, B., Woźniak, M., Damasevicius, R. Recommendation Based on Review Texts and Social Communities: A Hybrid Model. IEEE Access, 2019, 7, 40416-40427. https://doi.org/10.1109/ACCESS.2019.2897586
14. Kingma, D. P., Ba, J. L. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations, 2015.
15. Koren, Y. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, 426-434. https://doi.org/10.1145/1401890.1401944
16. Koren, Y., Bell, R., Volinsky, C. Matrix Factorization Techniques for Recommender Systems. Computer, 2009, 42(8), 30-37. https://doi.org/10.1109/MC.2009.263
17. Melville, P., Mooney, R. J., Nagarajan, R. Content-boosted Collaborative Filtering. Eighteenth National Conference on Artificial Intelligence, 2002.
18. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. Proceedings of the 26th International Conference on Neural Information Processing Systems, 2013, 2, 3111-3119.
19. Nisha, C. C., Mohan, A. A Social Recommender System Using Deep Architecture and Network Embedding. Applied Intelligence, 2018, 49(5), 1937-1953. https://doi.org/10.1007/s10489-018-1359-z
20. Rendle, S. Factorization Machines. IEEE International Conference on Data Mining (ICDM), 2010. https://doi.org/10.1109/ICDM.2010.127
21. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L. BPR: Bayesian Personalized Ranking from Implicit Feedback. Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, 2009, 452-461.
22. Sarwar, B., Karypis, G., Konstan, J., Riedl, J. Item-based Collaborative Filtering Recommendation Algorithms. Proceedings of the 10th International Conference on World Wide Web, 2001, 285-295. https://doi.org/10.1145/371920.372071
23. Wang, R., Cheng, H. K., Jiang, Y., Lou, J. A Novel Matrix Factorization Model for Recommendation with Lod-based Semantic Similarity Measure. Expert Systems with Applications, 2019, 123, 70-81. https://doi.org/10.1016/j.eswa.2019.01.036
24. Wang, X., He, X., Nie, L., Chua, T. S. Item Silk Road: Recommending Items from Information Domains to Social Users. Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, 185-194. https://doi.org/10.1145/3077136.3080771
25. 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. Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, 3119-3125. https://doi.org/10.24963/ijcai.2017/435
26. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R. S., Bengio, Y. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Proceedings of the 32nd International Conference on International Conference on Machine Learning, 2015, 37, 2048-2057.
27. Yang, B., Lei, Y., Liu, J., Li, W. Social Collaborative Filtering by Trust. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(8), 1633-1647. https://doi.org/10.1109/TPAMI.2016.2605085
28. Yang, J., Wang, H., Lv, Z., Wei, W., Song, H., E.-K., M., Kantarci, B., He, S. Multimedia Recommendation and Transmission System Based on Cloud Platform. Future Generation Computer Systems, 2016, 70, 94-103. https://doi.org/10.1016/j.future.2016.06.015
29. Yin, W., Schütze, Hinrich, Xiang, B., Zhou, B. Abcnn: Attention-based Convolutional Neural Network for Modeling Sentence Pairs. Transactions of the Association for Computational Linguistics, 2016, 4, 566-567. https://doi.org/10.1162/tacl_a_00244
30. You, Q., Jin, H., Wang, Z., Fang, C., Luo, J. Image Captioning with Semantic Attention. IEEE Conference on

Computer Vision and Pattern Recognition, 2016, 4651-4659. https://doi.org/10.1109/CVPR.2016.503

31. Yu, K., Schwaighofer, A., Tresp, V., Xu, X., Kriegel, H. Probabilistic Memory-based Collaborative Filtering. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(1), 56-69. https://doi.org/10.1109/TKDE.2004.1264822

32. Yu, Y., Wang, C., Hao, W., Yang, G. Attributes Coupling Based Matrix Factorization for Item Recommendation. Applied Intelligence, 2017, 46(3), 521-533. https://doi.org/10.1007/s10489-016-0841-8

33. Zanfir, M., Marinoiu, E., Sminchisescu, C. Spatio-temporal Attention Models for Grounded Video Captioning. Asian Conference on Computer Vision, 2016, 104-119. https://doi.org/10.1007/978-3-319-54190-7_7

34. Zhao, T., Mcauley, J., King, I. Leveraging Social Connections to Improve Personalized Ranking for Collaborative Filtering. Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, 2014, 261-270. https://doi.org/10.1145/2661829.2661998

35. Ziegler, C. N., Lausen, G., Schmidt-Thieme, L. Taxonomy-driven Computation of Product Recommendations. Proceedings of the 13th ACM International Conference on Information and Knowledge Management, 2004, 5, 406-415. https://doi.org/10.1145/1031171.1031252