An Approach for Item Recommendation Using Deep Neural Network Combined with the Bayesian Personalized Ranking

Zhongqin Bi[1], Siming Zhou[1], Xiaoxian Yang[2,3(✉)], Ping Zhou[1], and Jiale Wu[1]

[1] School of Computer Science and Technology, ShangHai University of Electric Power, Shanghai, China
[2] School of Computer and Information Engineering, Shanghai Polytechnic University, Shanghai, China
xxyang@sspu.edu.cn
[3] Shanghai Shang Da Hai Run Information System Co., Ltd., Shanghai, China

Abstract. This paper proposes a deep neural network model (SDAE-BPR) based on Stack Denoising Auto-Encoder and Bayesian Personalized Ranking for the problem of accurate product recommendation. First, we use the Stack Denoising Auto-Encoder (SDAE) as the input of the item's rating data and obtain the hidden features after encoding. Second, the Bayesian personalized Ranking (BPR) method is used to learn the hidden feature vector of the corresponding item. This model can avoid the influence of the sparseness of the matrix. Therefore, this model achieves the effect of more accurate recommendations of items. Third, to reduce the cost of model training, a unique pre-training and fine-tuning strategy is proposed in the deep neural network. Finally, based on the Movielens 20M dataset, the results of the SDAE-BPR, a traditional item-based collaborative filtering model and a user-based collaborative filtering model are compared. It is shown that the SDAE-BPR has higher accuracy. This method improves the accuracy of parameter estimation and the efficiency of model training.

Keywords: Recommendation · Stack Denoising Auto-Encoder · Bayesian Personalized Ranking · Deep learning · The sparseness of matrix

1 Introduction

The recommendation system plays an extremely important role in e-commerce platforms, because it helps the platform promote advertisements and products to users and leads to greater commercial benefits [1]. Currently, collaborative filtering is the most widely used commercial recommendation algorithm. This algorithm learns to build a rating matrix based on the existing item-user ratings to predict the user ratings of unknown items [2, 3]. With the advent of the era of big data, the number of users and products has soared. Most of the products have been rated by only a small number of users. Thus, the sparsity of the rating matrix seriously affects the quality of the recommendation results [4].

X. Wang et al. (Eds.): CollaborateCom 2019, LNICST 292, pp. 151–165, 2019.
https://doi.org/10.1007/978-3-030-30146-0_11

Therefore, to improve the accuracy, this paper proposes a new method. uses Stack Denoising Auto-Encoder based on Bayesian Personalized Ranking to determine the relevance ranking table for each unique item. This method is different from the previous method that relies on specific context information. For each item, we choose the automatic encoder method in the feature extraction step. This approach has the generalization capability due to adding the noise to the input data, achieving greater robustness [5]. Additionally, by ranking the similarity probabilities of other items and itself, it is guaranteed that these similar items are ranked higher than the dissimilar items, and this sorting method is proved to be effective and can solve the imbalance problem. To address the large computational cost, we proposed a pre-training + fine-tuning strategy [6] for the model.

In this paper, the proposed model integrates the advantages of the BPR and the SDAE into a deep learning model. Compared with the traditional collaborative filtering recommendation algorithm, this model has some unique advantages. First, the rating vector of each item can obtain a more complex representation of the hidden features after extracting the deep network through the SDAE. Meanwhile, the addition of noise also improves the anti-interference property of the model, making the extracted features more reliable. Second, the final BPR ranking part can better capture the unique characteristics of each item and give the probability of the similarity between each item to reduce the impact of data sparsity effectively. Thus, this approach helps to improve the accuracy of recommendation. Third, in order to avoid poor parameter estimation, we design a pre-training and fine-tuning strategy based on the Bernoulli probability model. In the last part of this paper, the experiments carried out on real commodities datasets are described. The results show that this model obtains more accurate recommendation results than the classical collaborative filtering algorithm. Figure 1 shows an overview of our method.

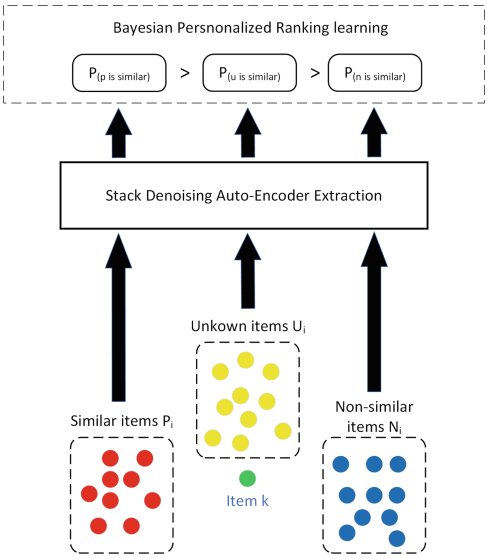

Fig. 1. Overview

The remainder of this paper is organized as follows. In Sect. 2, we introduce the deep neural network model based on Stack Denoising Auto-Encoder and Bayesian Personalized Ranking. Section 3 presents the details and results of the experiments. Section 4 reviews the related works. Section 5 states the conclusions and describes future research directions.

2 Model Framework

This section first describes a pairwise raking task in the commodity recommendation problem. Then, we propose a Bayesian deep neural network model based on the BPR and describe the pre-training strategy. Figure 2 shows the architecture of the model.

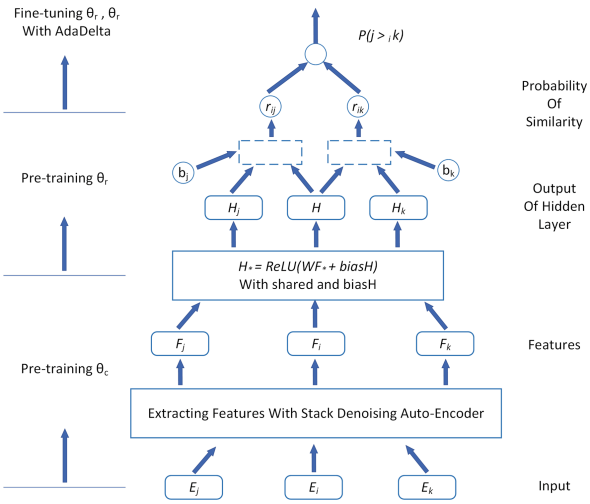

Fig. 2. SDAE-BPR

2.1 Preliminaries

User $u \in R^N$, item $I \in R^I$. $E \in \{0,1\}^{I \times N}$ is the rating matrix formed by all of the user's scores on all of the items. Notation $e_{iu} = E(i,u) = 1$ indicates that user u is interested in item i, and $e_{iu} = E(i,u) = 1$ indicates that user u is not interested in item i. In most e-commerce platforms, the rating matrix E is usually sparse because most users can only obtain access to a small part of the entire item database.

Define a similarity probability matrix $R \in [0,1]^{I \times I}$, with notation $r_{ij} = R(i,j)$ representing the similarity probability of items i and j. Thus, for each item i, it can be divided into two disjoint sets that include the set of items with similar relations $P_i = \{j | r_{ij} = 1\}$ and the set of items with uncertain similar relations $M_i = \{j | r_{ij} < 1\}$. Here, we seek to recommend the ranking task learning model. This model ensures that all of the items with a similar relationship are in front of the of missing items. We can also divide the missing items M_i into the unknown U_i and dissimilar N_i, implying

$M_i = U_i \cup N_i$. In the training process of ranking tasks, $j \in P_i$ and $k \in M_i$, or $j \in U_i$ and $k \in N_i$ should be guaranteed, and the probability of item similarity r_{ij} should be greater than r_{ik}. In Bayesian personalized sorting, this relation is called partial relation $j >_i k$.

Based on the above definition, the product recommendations in this paper can be divided into two types of partial relations: $\{j >_i k | j \in P_i \cup k \in M_i\}$ and $\{j >_i k | j \in U_i \cup k \in N_i\}$. The set of all partial relations for item i is expressed as $R_i = \{(j,k) | j >_i k\}$. Therefore, it is observed that the ranking task is the essence of the product recommendation task in this paper. Compared with the classification and fitting, the sorting task can better avoid the imbalance problem. The final goal of this ranking task is to maximize the likelihood probability of the ranking given by:

$$\max \prod_{i \in R^I} \prod_{(j,k) \in R_i} P(j >_i k) \tag{1}$$

2.2 Feature Extraction

The SDAE is a deep structure model that connects multiple Denosing Auto-Encoders. For a common Auto-Encoder, it is easy to obtain an identity function if the features of the rating vectors are extracted only by minimizing the error between the input and output. The Auto-Encoders are linked together, rather than maintained as single Auto-Encoders. The reason is that if the noise is added to the user rating of each item, the encoder used in refactoring the input data must be forced to remove the noise. After training, due to the process of noise removal, the feature extraction layer will obtain a function that is more complex than the identity function. To eliminate the influence of noise in the ratings, we add the L2 regularization term [7] into the loss function. This term penalizes an excessively large weight. In addition, due to the background of the massive data available on the Internet, the shallow model has limited ability to express numerous rating vectors and cannot accurately distinguish the characteristics of different items. By contrast, the deep model can obtain the hidden characteristics behind each item's rating due to its more powerful deep extraction ability. The deep model can extract results more vividly and representatively than the shallow model. The design of the SDAE is described in Table 1.

Table 1. Algorithm of SDAE for feature extraction

Algorithm 1 SDAE for Feature Extraction
Input: Rating vector x
Output: Extracted feature F_i
Initialization: Randomly generate $W^{(1)}$, $b^{(1)}$, $\nabla J(\theta)$, $f^0 = x$, $h = 1$
1: while $\nabla J(\theta)$!= 0
2: for $h < 3$
3: Calculate result f^h by taking f^{h-1} through h layer
4: Update $W^{(h)}$, $b^{(h)}$, by BP method in h layer
5: $h = h + 1$
6: Calculate $\nabla J(\theta)$ by equation (3)
7: $F_i = f^h$
8: return F_i

In Table 1, $H = 3$ layers. The structure of the SDAE is *U-A-B*. Here, *U* is the input layer of each rating vector. *A* and *B* are the first hidden layer and the second hidden layer, respectively. For the original rating x, assign value 0 to some of the data before inputting into the network to obtain $x^{\wedge}$ in proportion. Then, $x^{\wedge}$ is input, and a greedy strategy is used in the training network of each layer step by step. All of the above steps comprise the pre-training of the SDAE layer. Finally, we use the results of the last layer as hidden characteristics. Considering that the input data are nonlinear and negative, it will be better to choose a sigmoid function as the activation function. Here, W is the weight matrix, and B is the bias vector. The training process is as follows: first, to obtain f^1, $x^{\wedge}$ is used as the input into the first hidden layer. Then, put f^1 into the decoding function in order to obtain $x'^{(1)}$. For each rating of the items, its reconstruction error function in any layer is given by Eq. (2).

$$l(x, x') = -\sum_{n=1}^{N} x_n \log(x') + (1 - x_n) \log(1 - x'_n) \tag{2}$$

For the whole training set with I items, the error function of the integration in this layer is given by Eq. (3).

$$J(\theta) = -\frac{1}{I}\left[\sum_{i=1}^{I}\sum_{n=1}^{N} x_n^{(i)} \log(x_n'^{(i)}) + (1 - x_n^{(i)}) \log(1 - x_n'^{(i)}) + \frac{\lambda}{2N}\sum_{h=1}^{H-1}\sum_{i=1}^{I_h}\sum_{n=1}^{I_{h+1}} (\theta_{ni}^{(h)})^2\right] \tag{3}$$

The goal of training is to minimize the error function in each step of the iteration. Therefore, to prevent over-fitting, the regularization parameter λ is introduced to avoid the weight becoming too large. For the error function of each layer, the Back-Propagation method and the Stochastic Gradient Descent method are combined to obtain the parameter θ^l of this layer. Under this parameter, the output f^l of this layer is the input of the hidden layer of the next layer. Repeat the above training process and keep the parameters of each training layer.

2.3 Learning to Rank

The rest of the deep model proposed in this paper is based on the hidden layer H. In the final output layer, the BPR is adopted to rank and learn an output so that the most similar items are ranked at the top, and the final recommended items are selected from the top k items. The training model maximizes the likelihood probability specified by Notation (1), and the loss function of the whole model is given by Eq. (4).

$$L(\theta_c, \theta_r) = -\sum_{i}\sum_{(j,k\in R')} P(j >_i k) + \lambda_1 \|\theta_c\|^2 + \lambda_2 \|\theta_r\|^2 \tag{4}$$

Here, $\theta_c = \{W_1^1, W_1^2, b_1^1, b_1^2\}$ is the weight and bias of the SDAE part. $\theta_r = \{W_2, b_2, b_3\}$ is the parameter of the other parts. After the pre-training in the SDAE layer, θ_c and θ_r were determined again by the Backward Propagation Stochastic Gradient Descent method. When the gradient is stable, the probability that all other items are

similar to itself is calculated. Then, a ranking list can be created easily. Then, the system recommends items to users according to this list. After the training, the user rating vector of any two items is used as input, and the probability value of similarity $r_{ij} = P(e_{ij} = 1)$ between two items is determined. This probability value is the reason why we recommend these items.

Hidden Layer. The input part is F_i, F_j, F_k, which is the feature extracted by i j, k through the SDAE. The purpose of the hidden layers is to embed them in H_i, H_j, H_k for further calculation. In this layer of the network, we select the ReLU function as the activation function. The hidden layer effect here is not to extract features, so we choose the ReLU function with relatively less information but faster convergence Eq. (5).

$$H_i = ReLU(W^2 F_i + b_2) \tag{5}$$

Predicting Layer. The input is H_i, H_j, H_k, the output is r_{ij} and r_{ik}, and the activation function is given by Eq. (6)

$$r_{ij} = \sigma(b_3^i + b_3^j + H_j H_i^T) \tag{6}$$

Sigmoid is chosen as the activation function because the probability of the final output should be within [0, 1]. In the previous hidden layer, to improve the efficiency of training, all of the items use the same parameter $\{W_2, b_2\}$, but each item has its own unique parameter b_3^i in the predicting layer. Therefore, it is more likely to explore the inherent potential of each item and to improve the accuracy of the recommendation. The probability of the partial relation between j and k is defined by Eq. (7).

$$P(j >_i k) = \frac{r_{ij} - r_{ik}}{2} + 0.5 \tag{7}$$

2.4 Pre-training of θ_r and Fine-Tuning

In the feature extraction section above, the pre-training method of θ_c was described. Here, the pre-training method of θ_r and the fine-tuning method of the whole model are mainly introduced. Table 2 describes the flow of these two algorithms. Later, we will provide a detailed explanation of how each step in the algorithms is implemented in combination with this chart.

Pre-training θ_r. The feature F_i generated after training in the part of the SDAE is used as the input, and the output is r_{ij}, r_{ik}. To estimate the parameter, set $\theta_r = \{W_2, b_2, b_3\}$ of project i, and the remaining structural parts must be pre-trained. The similarity relation e_{ij} between the items in the training set is regarded as a sample of the Bernoulli distribution with a parameter r_{ui} given by Eq. (8).

$$p(e_{ij}|r_{ij}) = r_{ij}^{e_{ij}}(1 - r_{ij})^{1-e_{ij}} \tag{8}$$

The likelihood probability corresponding to the above equation is defined as follows:

$$L = \sum_{i,j} (e_{ij} \log r_{ij} + (1 - e_{ij}) \log(1 - r_{ij})) \tag{9}$$

To estimate θ_r, let us define a function $r_{ui} = \mathrm{g}\ (F_i, F_j)$ that goes from F_i, F_j to r_{ui}. Item i and item j are sampled from the positive example set P_i and the negative example set N_i, respectively. The negative examples are collected from set N_i rather than from U_i because this approach can greatly improve the training efficiency. Therefore, the logarithmic likelihood probability θ_r is defined using Eq. (10)

$$L(\theta_r) = \sum_{i} \sum_{i \in P_i} \log g(F_i, F_j) + \sum_{j \in N_i} \log[1 - g(F_i, F_j)] - \lambda ||\theta_r||^2 \tag{10}$$

For the above equation, we use the Stochastic Gradient Descent optimization. In each iteration of the SGD, the updating method of θ_r is given by Eq. (11), where η is the learning rate and λ is the regularization parameter.

$$\Delta\theta_r = \eta \cdot ([e_{ij} - g(F_i, F_j) \cdot \frac{\partial g}{\partial \theta_r}] - \lambda\theta_r) \tag{11}$$

Fine-Tuning. Fine-tuning is necessary if the pre-training parameters are separately trained. To give the whole entire parameter a better initial space, we adopted the AdaDelta algorithm [8] that is based on the history of the gradient and weight to scale the SGD learning rate and can accelerate the convergence speed of the neural network in the first stage of the training process.

Table 2. Algorithm of pre-training θ_r and fine-tuning.

Algorithm 2 Pre-training θ_r and Fine-tuning
Initialization: Randomly generate θ_r
1: repeat (Pre-training θ_r)
2: for all $i, j \in$ batch
3: Calculate r_{ij} from F_i and F_j
4: Back propagation through SDAE-BPR
5: until Convergence
6: for Epoch in AdaDelta (Fine-tuning)
7: Random select a partial relation i, j, k
8: Calculate r_{ij}, r_{ik} from E_i, E_j, E_k
9: Update SDAE-BPR with AdaDelta
10: end

3 Experiments and Analysis

3.1 Data Sets and Evaluation Indicators

In the experimental part of this work, the proposed model is compared with some existing classical collaborative filtering recommendation algorithms. The following experiments are based on the movielens 20M dataset and movielens 1M dataset, respectively. For each method, 10-fold cross-validation is performed on the data set, and the average result is shown at the end.

To measure the performance of the different methods on the specified data sets, we use AUC as an indicator to evaluate the performance of different recommended methods and draw Precision-Recall graphs for intuitive comparison.

Movielens 20M Dataset [9]. This dataset is a stable benchmark dataset. A total of 238,000 users made 27,000 comments on 27,000 movies. These 27,000 movies come with attribute tags and 12 million movie correlation scores.

Movielens 1M Dataset. This dataset is a small dataset with 600 users applying 100,000 ratings and 3,600 attribute tags to 9,000 movies.

UB – CF [10]. User-based collaborative filtering is a collection of similar users based on the user's rating of the item that measures the similarity between the users, and then organizes them into a sorted catalogue for recommendation based on their favourite items. However, due to the large number of items in the Internet, most users only evaluate a few items, so there is a problem of sparsity that is difficult to solve.

IB – CF [11]. Item-based collaborative filtering uses the interactive information between the users and items to make recommendations for the users. Currently, IB-CF is the most widely used recommendation algorithm. However, the adopted shallow model cannot learn the deep features of users and items.

AUC [12]. Formally, AUC considers the ranking quality of sample prediction, while the ROC curve represents the comparison between the TPR and FPR as the classification threshold standard changes. Therefore, the AUC area is used here instead of the ROC curve as the measurement index. AUC values range from 0.5 to 1, with higher values indicating better performance. To evaluate the recommendation performance, we use the AUC between P and U to measure the model's ability to rank.

Precision-Recall Curve [13]. We can rank the samples according to the prediction result of the learner. The samples in the first place are the examples that the learner considers "most likely" to be positive, and the samples in the last place are the examples that the learner considers "least likely" to be positive. By taking samples as the positive examples one by one in this order for prediction, the current accuracy and recall can be calculated each time. Equations (12) and (13) are formula definitions, where TP, FP, FN represents true positive samples, false positive samples, and false negative samples, respectively. The Precision-Recall curve is obtained by plotting the accuracy ratio on the vertical axis and the recall ratio on the horizontal axis. If the Precision-Recall curve of one learner is completely wrapped by the curve of another learner, the latter can be asserted to have better performance than the former.

$$P = \frac{TP}{TP + FP} \tag{12}$$

$$R = \frac{TP}{TP + FN} \tag{13}$$

In order to evaluate the ranking quality of the recommendation results, we also adopted NDCG@n as one of the recommendation indicators.

NDCG [14]. Normalized Discounted Cumulative Gain (NDCG) is the ratio of the DCG to the described Ideal DCG, which means its similar items P are always ranked before the rest of the items. The higher NDCG value indicates a better learning performance. Commonly, the NDCG@n that calculates the NDCG result over the top ranked n items are used in the recommendation tasks. The NDCG result is described as follows:

$$N(n) = Z_n \sum_{j=1}^{n} (2^{r(j)} - 1) / \log(1 + j) \tag{14}$$

In formula 14, j represents the number of goals we want in the results and Z_n represents the normalization. In our experiments, we calculate NDCG@5 for each item and average them as a metric. Meanwhile, we also try different n for detailed estimation.

3.2 Result Analysis

To measure the performance of these models on datasets of different sizes, these models are validated using the movielens 20M dataset and the movielens 1M dataset, respectively. This validation is performed by calculating the AUC of the results and drawing P-R graphics to compare the comprehensive performance of the model. It was found that the AUC of the SDAE-BPR and the P-R curve are is higher than those of the classical algorithms. At the same time, the NDCG@n of the SDAE-BPR is also larger than those for the other two classical collaborative filtering algorithms.

Table 3. Comparison of AUC and NDCG@6 results.

	Movielens 20M		Movielens 1M	
	AUC	NDCG@6	AUC	NDCG@6
UB-CF	0.926	0.883	0.913	0.839
IB-CF	0.941	0.892	0.927	0.874
SDAE-BPR	0.967	0.971	0.944	0.958

AUC. As observed from the examination of the data presented in Table 3, the AUC of the SDAE-BPR is 2.6% higher than that of the IB-CF. The AUC of the SDAE-BPR is also 4.1% higher than that of UB-CF for the movielens 20M dataset. For the movielens 1M dataset, the AUC of SDAE-BPR was 1.7% higher than that of IB-CF and 3.1% higher than that of UB-CF. Based on the horizontal comparison, with the increase in the size of the training samples, the model can fit better. Therefore, all of these three methods perform better for large datasets than for small datasets. From a

longitudinal perspective, the SDAE-BPR has the best performance in each training set followed by the IB-CF and the UB-CF. UB-CF shows the worst performance because when the number of items is too large, each user can only evaluate a few items. Thus, it is difficult to find enough users who are very similar in the training set. Because it measures the similarity between the items, the IB-CF can find many similar items in the training set. In addition to calculating the similarity of the items, the SDAE-BPR also deeply extracted the user evaluation vector to obtain the unique characteristics of each item. This deep extraction ensures the higher accuracy of the recommendation results. Furthermore, we can easily find that the difference in AUC is larger than that in NDCG@6. Here, we can assume that the main function of the SDAE-BPR is to rank a better result.

Precision-Recall Curve. In Fig. 3, the P-R curve of SDAE-BPR basically wraps the curves of the other two models. This finding is also observed in Fig. 4. According to the definition of the P-R curve in the previous paper, we can conclude that the comprehensive performance of the SDAE-BPR is the best in both large and small data sets. The conclusions drawn here are in accordance with the results of the AUC. By comparing these two results, we can clearly determine that the SDAE-BPR has better comprehensive performance than the classical collaborative filtering algorithm. However, in Fig. 4, the P-R curve of all methods nearly overlap. Even with the increasing amount of training data sets, in Fig. 3, their Precision-Recall curves were not becoming sufficiently different. Based on this phenomenon, we infer that significantly improving the performance of the model is not the advantage of the SDAE-BPR. To verify this assumption, analysis of the NDCG must be performed as described in the next section.

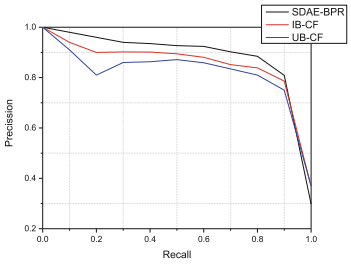

Fig. 3. Precision-Recall curves on movielens 20M

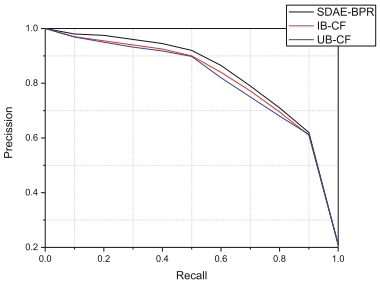

Fig. 4. Precision-Recall curves on movielens 1M

NDCG@n. First, we need to choose a value of n, that is, to ensure that the NDCG can be as large as possible, while the training cost is as small as possible. It is observed from Fig. 5 that the NDCG will be stable when n equals *6*. This condition is also observed for the results presented in Fig. 6. Therefore, we choose *6* as the value of n. From Table 2, in each dataset, the NDCG@6 is much larger than other two classical collaborate filtering algorithms. The curve of the SDAE-BPR in Fig. 5 is much higher than the other two curves. In Fig. 6, the differences still stay the same. This result means the SDAE-BPR always has more accurate rank results than the classic methods. Therefore, the SDAE-BPR has the highest rank quality among these three algorithms. In addition, the difference of the NDCG@6 among all methods is also much larger than that of the AUC. This result is because the goal of the SDAE-BPR in training is to maximize the difference between the positive example probability and the negative example probability, rather than simply to calculate the similarity of each item after fitting. It was found that the SDAE-BPR model is more suitable for a small number of precise recommendation application scenarios. In addition, for each different item i, the bias b_i belonging to this item is added, also contributing to improving the quality and accuracy of the recommendation ranking results.

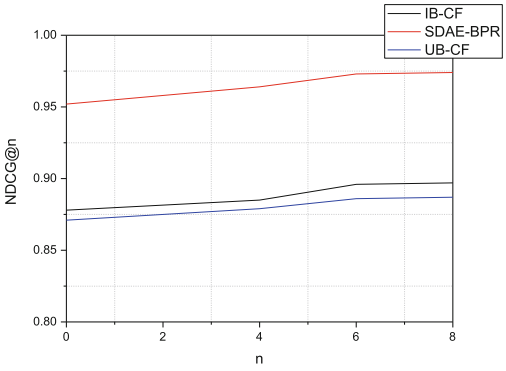

Fig. 5. NDCG@n on movielens 20M

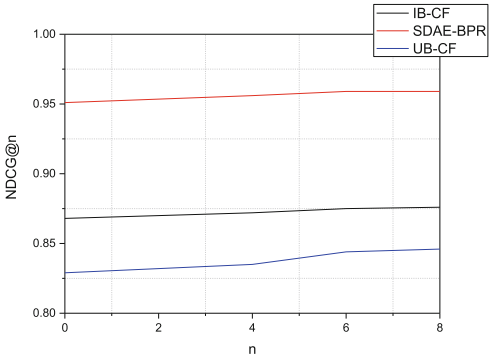

Fig. 6. NDCG@n on movielens 1M

4 Related Work

The most popular model for the recommender system is k-nearest neighbour (kNN) collaborative filtering. [15] Recently, matrix factorization (MF) has become very popular in recommender systems both for implicit and explicit feedback. In early work, [16] singular value decomposition (SVD) has been proposed to learn the feature metrics. The MF models learned by SVD have been shown to be highly prone to overfitting. Below, we review some of the better methods mentioned in this paper.

Deep Learning. Deep learning has become highly popular on the Internet for big data and artificial intelligence [17]. Deep learning, by combining low-level features to form denser high-level semantic abstracts, can automatically discover the distributed feature representation of data. Deep learning can solve the problem of manual design features in traditional machine learning and has achieved breakthroughs in image recognition, machine translation, speech recognition, online advertising and other fields. In the field of image recognition, the accuracy rate of deep learning exceeded 97% in the 2016 ImageNet image classification competition. In the field of machine translation, the Google neural machine translation system (GNMT) based on deep learning has achieved a translation level close to that of humans in the field of English to Spanish and English to French [18]. In the field of online advertising, deep learning is widely used to predict the click rate of advertisements and has achieved great success in its application by Google [19], Microsoft [20], Huawei [21], Alibaba [22] and other enterprises. Deep learning involves a wide range of machine learning technologies and structures. The SDAE used in this paper belongs to this deep learning structure.

SDAE. Auto-Encoder is a common method used in feature extraction by neural network [23, 24]. These networks are trained to reconstruct their inputs by dimensionality reduction, resulting in better characterization than the original data. Common methods for extracting features of neural networks include the Convolutional Neural Network (CNN) [25] and Recurrent Neural Network (RNN) [26]. However, the CNN is a kind of multi-layer perceptron, which is mainly used to process two-dimensional or three-dimensional image data. Additionally, the nodes between each hidden layer of the RNN are connected and able to memorize the past information, so that this method is more suitable for sequence modelling. Considering that the scoring data in the recommendation system are all one-dimensional values without sequence relations, methods such as the CNN and RNN cannot play a meaningful role in feature extraction but will increase the computational complexity. Therefore, the stack denoising auto-encoder may be a better choice [27]. The advantage of stack structure [28, 29] is that multi-hierarchy abstract data representation and deeper implicit data representation can be obtained by stacking multiple automatic encoders.

BPR. From an algorithmic point of view, the existing methods of recommending problems can be approximately divided into three categories: classification, fitting, and ranking. [30, 31] The classification method can be regarded as a binary classification problem, using the predefined features to train the classifier, and finally using the classifier to predict the similarities between the items. The method of fitting is to convert the scores of items into a real value rating matrix and to use a collaborative filtering method such as matrix decomposition to predict the similar probability of

items without scoring. The problem of data sparsity will make classification or fitting methods be biased towards dissimilarity. [32] The sorting method considers the recommendation as a sort of learning task. For each item, by ranking the similarity probabilities of other items and itself, it is guaranteed that these similar items are ranked higher than the dissimilar items, and this sorting method proved to be effective and can solve the imbalance problem. Among these models, the BPR model [33, 34] defines the Bayesian pairwise ranking relationship between the items. The relationship is that probabilities of similar items should be greater than those of the non-similar items. This model has been verified to achieve good performance.

5 Conclusions

In this paper, the existing BPR model and the SDAE are combined to recommend products. The SDAE is used to extract the implicit characteristics of the user evaluation vector, and the already extracted BPR is based, in part, on deep hidden features to obtain the features of the products and, on the basis of the entire model, to propose a set that is suitable for the preliminary training of the model; then, an optimization strategy is used to speed up the training efficiency and improve the recommendation accuracy. As demonstrated in the experimental verification, the model proposed in this paper has better performance than the existing classical collaborative filtering commodity recommendation algorithm, obtaining higher accuracy and better sorting of the results and avoiding the impact of sparsity. However, this model still has some shortcomings. For example, when the data volume is large, features are extracted for each item, and the similarity probability of any two items must be calculated. The model learning and data preservation may encounter some difficulties. However, the application of deep learning in the recommendation system has been studied by many researchers and has been proven to be feasible. Therefore, we believe that this research direction is promising.

In the future, we will study the interpretability of the algorithm. Currently, we can only provide the answer about probability to the user. It is not yet possible to convincingly show why the probability should have this value. Furthermore, the users' interest in different goods on e-commerce platforms changes rapidly with time. It will be useful to connect our methods to the changes in user interest.

Acknowledgments. This paper is supported by the Youth Foundation of Shanghai Polytechnic University under Grant No. EGD18XQD01; the CERNET Innovation Project No. NGII2017 0513.

References

1. Xie, F., Chen, Z., Xu, H., et al.: Threshold based similarity transitivity method in collaborative filtering with cloud computing. Tsinghua Sci. Technol. **18**(3), 318–327 (2013)
2. Stanford University's machine learning course on courser. https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filterin. Accessed 29 Mar 2019

3. Li, W., Yeung, D., Zhang, Z.: Generalized latent factor models for social network analysis. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Spain, pp. 1705–1710 (2011)
4. Bobadilla, J., Ortega, F., Hernando, A., et al.: Recommender systems survey. Knowl.-Based Syst. **46**, 109–132 (2013)
5. Vincent, P., Larochelle, H., Bengio, Y., et al.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, pp. 1096–1103 (2008)
6. Ding, D., Zhang, M, Li, S., et al.: BayDNN: friend recommendation with bayesian personalized ranking deep neural network. In: ACM (2017)
7. Wang, H., Shi, X., Yeung, D.: Relational stacked denoising autoencoder for tag recommendation. In: Proceedings of the 29th Conference on Artificial Intelligence, Austin, USA, pp. 3052–3058 (2015)
8. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. Comput. Sci. (2012)
9. Recommend for new research. http://grouplens.org/datasets/movielens. Accessed 30 Mar 2019
10. Zhao, Z., Shang, M.: User-based collaborative-filtering recommendation algorithms on Hadoop. In: WKDD 2010 Third International Conference on IEEE, pp. 478–481 (2010)
11. Su, X., Khoshgoftaar, T.: A survey of collaborative filtering techniques. Adv. Artif. Intell. **2009**(4), 421425:1–421425:19 (2009)
12. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recogn. **30**(7), 1145–1159 (1997)
13. Boyd, K., Eng, Kevin H., Page, C.D.: Area under the precision-recall curve: point estimates and confidence intervals. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8190, pp. 451–466. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40994-3_29
14. Yilmaz, E., Kanoulas, E., Aslam, J.: A simple and efficient sampling method for estimating AP and NDCG In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 603–610. ACM (2008)
15. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Trans. Inform. Syst. **22**(1), 143–177 (2004)
16. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Incremental singular value decomposition algorithms for highly scalable recommender systems. In: Proceedings of the 5th International Conference in Computers and Information Technology (2002)
17. Silver, D., Huang, A., Maddison, C., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
18. Wu, Y., Schuster, M., Chen, Z., et al.: Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1708.05123 (2017)
19. Cheng, H., Koc, L., Harmsen, J., et al.: Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender System, Boston, USA, pp. 7–10 (2017)
20. Shan, Y., Hoens, T., Jiao, J., et al.: Deep crossing: web-scale modeling without manually crafted combinatorial features. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, USA, pp. 255–262 (2017)
21. Guo, H., Tang, R., Ye, Y., et al.: DeepFM: A factorization-machine based neural network for CTR prediction. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, pp. 1725–1731 (2017)
22. Zhou, G., Song, C., Zhu, X., et al.: Deep interest network for click-through rate prediction. arXiv preprint arXiv:1708.05123 (2017)

23. Wu, Y., DuBois, C., Zheng, A., et al.: Collaborative denoising auto-encoders for top-n recommender systems. In: Proceedings of the 9th ACM International Conference on Web Search and Data Mining, San Francisco, USA, pp. 153–162 (2016)
24. Vasile, F., Smirnova, E., Conneau, A.: Meta-Prod2Vec: product embeddings using side-information for recommendation. In: Proceedings of the ACM Conference on Recommender Systems, Boston, USA, pp. 225–232 (2016)
25. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. In: Proceedings of the Advance in Neural Information Processing Systems, Lake Tahoe, USA, pp. 1097–1105 (2012)
26. Graves, A., Jaitly, N.: Towards end-to-end speech recognition with recurrent neural networks. In: Proceedings of the 31st International Conference on Machine Learning, Beijing, China, pp. 1764–1772 (2014)
27. Strub, F., Mary, J.: Collaborative filtering with stacked denoising AutoEncoders and sparse inputs. In: Proceedings of the NIPS Workshop on Machine Learning for E-Commerce, Montreal, Canada (2015)
28. Vincent, P., Larochelle, H., Lajoie, I., et al.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**(12), 3371–3408 (2010)
29. Bengio, Y., Lamblin, P., Popvici, D., et al.: Greedy layer-wise, training of deep networks. In: Proceedings of the Advances in Neural Information Processing Systems, Vancouver, Canada, pp. 153–160 (2007)
30. Li, Z., Fang, X., Sheng, O.R.L.: A survey of link recommendation for social networks: methods, theoretical foundations, and future research directions. Comput. Sci. (2015)
31. Han, S., Yan, X.: Friend recommendation of microblog in classification framework: using multiple social behavior features. In: International Conference on Behavior, Economic and Social Computing, pp. 1–6 (2014)
32. Jeni, L., Cohn, J., Torre, F.: Facing imbalanced data recommendations for the use of performance metrics. In: Humaine Association Conference on Affective Computing and Intelligent Interaction, pp. 245–251 (2012)
33. Qiu, S., et al.: Item group based pairwise preference learning for personalized ranking. In: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1219–1222 (2014)
34. Rendle, S., Freudenthaler, C., Gantner, Z., et al.: BPR: Bayesian personalized ranking from implicit feedback In: Conference on Uncertainty in Artificial Intelligence. AUAI Press (2009)