DEEPAUTOENCF: A DENOISING AUTOENCODER FOR RECOMMENDER SYSTEMS

BHAKTI AHIRWADKAR
Department of Computer Science and Engineering,
Marathwada Institute of Technology, Aurangabad, Maharashtra, INDIA.
Email: bhaktipr@gmail.com

SACHIN N. DESHMUKH
Department of Computer Science and IT,
Dr. Babasaheb Ambedkar Marathwada University, Aurangabad, Maharashtra, INDIA.
Email: sndeshmukh@yahoo.com

Abstract - Recommender Systems are software techniques which can be used to filter out data from the volumes of data available online and provide recommendations to users in their area of interest. These techniques use information related to users and items in addition to the ratings given by users to various items or providing recommendations. In the last two decades, deep learning techniques have shown promising results in various areas of computer vision, video recognition, natural language processing etc. These techniques have been used for recommender systems in recent years and have shown improvement in performance. In this paper we propose a model, DeepAutoEnCF, that uses Denoising Autoencoder for predicting user ratings. It uses dropout for regularizing the model and adding noise to input for prediction of ratings. The model uses side information along with unique additional information for improving the performance.

Keywords : Recommender Systems; Collaborative Filtering; Content Based Collaborative Filtering; Hybrid Systems; Memory Based Approach; Model Based Approach; Deep Learning ; Autoencoders; Dropout.

1. Introduction

The widespread use of internet has made it easy for people to get information about many things they find interesting. But, due to the increase in online data day by day, it is getting difficult to extract useful information from such voluminous data. Recommender Systems (RS) play important role in assisting users to get information about things of their interest. Whenever users visit website for say online purchasing, they provide ratings to the items purchased. RSs are software that use these ratings provided by users and any other additional information about users and items to provide suggestions whenever users visit the website again. The task of recommender systems is therefore to predict the ratings of items to which the user has not rated and use these predicted ratings for recommending new items to users. The applications of recommender systems are in many areas like entertainment, service industry, online shopping, news article recommendations etc.

Traditional Recommender Systems are classified into Content Based Recommendation System, Collaborative Filtering and Hybrid Systems [Sarwar *et al.*, (2000)]. Content based recommender systems use features of items that were previously rated and liked by the user and try to recommend similar items to the user. Here, the system uses only the current user's history for building the profile of the user. Collaborative Filtering (CF) recommendation technique on the other hand tries to recommend those items to a user, that other similar users have liked in the past. Here, the system first finds set of similar users/items for the current user/item and then recommends those items that this set of users had liked earlier. Hybrid Recommender Systems are a combination of two or more content based and collaborative filtering techniques.

1.1 Collaborative filtering systems

Collaborative filtering systems use likings of other users for recommending new items to current user [Sarwar *et al.*, (2000)], [Isinkaye et al., (2015)]. For this task is uses two approaches: Memory based CF and Model based CF.

1.1.1 Memory based approach

This approach uses ratings provided by the user for calculating similar set of users for current user (User Based CF) or similar set of items for current item (Item based CF) [Ekstrand *et al.*, (2010)]. For predicting missing ratings, it uses weighted sum over this set of similar users/items. For calculating similarity commonly used measures are Pearson correlation of vector cosine similarity [Isinkaye et al., (2015)]. The Pearson correlation between users a and b is given in Eq. (1):

$$W_{a,b} = \frac{\sum i\epsilon I(r_{a,i}-\bar{r}_a)-(r_{b,i}-\bar{r}_b)}{\sqrt{\sum i\epsilon I(r_{a,i}-\bar{r}_a)^2}\sqrt{\sum i\epsilon I(r_{b,i}-\bar{r}_b)^2}} \quad (1)$$

where $\bar{r}_a$ is the average rating of co-rated items of the a[th] user. For two items i and j rated by users u∈U, the similarity is given in Eq. (2):

$$W_{i,j} = \frac{\sum u\epsilon U(r_{u,i}-\bar{r}_i)-(r_{u,j}-\bar{r}_j)}{\sqrt{\sum u\epsilon U(r_{u,i}-\bar{r}_i)^2}\sqrt{\sum u\epsilon U(r_{u,j}-\bar{r}_j)^2}} \quad (2)$$

where $r_{u,i}$ is the rating given by user u on item i, $\bar{r}_i$ is the average rating of the i[th] item by the users in U. Eq. (3) gives the vector cosine similarity between two items or users i and j which is the dot product of the two vectors representing them [Sarwar *et al.*, (2000)]:

$$W_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \bullet \vec{j}}{||\vec{i}||*||\vec{j}||} \quad (3)$$

The prediction of rating for user u for an item i for user based CF is done with weighted sum [Sarwar *et al.*, (2000)], [Su and Khoshgoftaar, (2009)] over set of similar users V who have rated the item i:

$$P_{u,j} = \bar{r}_u + \frac{\sum v\epsilon V(r_{v,i}-\bar{r}_v).w_{u,v}}{\sum_{v\epsilon V}|w_{u,v}|} \quad (4)$$

where $\bar{r}_u$and $\bar{r}_v$ are the average ratings for the users u and v and $w_{u,v}$ is the weight between user u and item v.

1.1.2 Model based approach

This approach builds a model to learn hidden relationships between the data for predicting missing ratings [Isinkaye et al., (2015)]; [Ekstrand *et al.*, (2010)]. Many algorithms have been developed with this approach for collaborative filtering like Bayesian models, clustering, dependency networks etc. Most popular model based approaches remain the approaches based on Matrix Factorization that represent ratings as inner product of user and item factor vectors obtained by mapping them to a joint latent factor space [Koren, (2009)].

If the ratings provided by the users are sparse, the memory based approaches may not perform well. Also, they suffer from cold start problem which is; they cannot recommend new items without ratings to users or give suggestions to new users who have not rated. Scalability is also an issue with these approaches. To overcome these problems model based approaches have been developed which try to learn relations from the data and handle the scalability problems with clustering. These models are linear and fail to learn complex relations between the data. Due to this in last decade deep learning approaches have been used for recommender systems [Zhang *et al.*, (2017)]. Various regularization techniques have been used for reducing the over fitting of the model. Dropout can be used for regularization which can perform better than other techniques. In this paper we propose a model for collaborative filtering, DeepAutoEnCF based on Denoising Autoencoders (DAE) for rating predictions. We try to improve the performance of the model by using dropout at hidden layers for regularizing the model and at input layer for adding noise to the Autoencoder. To improve the performance, the model uses the user and item additional information available in the dataset. It also uses unique additional information for users which is calculated using user ratings and movie genres.

2. Deep Learning For CF

In recent years, Deep Learning (DL) based approaches have been used in almost all areas for research including Image Processing, Natural Language Processing (NLP), Speech Recognition and many others. They have been used for Recommender Systems also and have shown promising results. Most common approaches are Multi Layer Perceptron, Convolutional Neural Networks, Autoencoders and Recurrent Neural Networks and Restricted Boltzman Machine [Betru *et al.*, (2017)]. For RSs deep learning approaches have been used either independently or are integrated with traditional approaches.

2.1 Autoencoders

Autoencoders are unsupervised neural networks that are trained to reconstruct the given input x as its output. For achieving this, it first encodes the input into short code and then decodes it to reconstruct the output. In a simple Autoencoder with one hidden layer, the input x ϵR^D is fed to the hidden layer, also called as bottleneck of the Autoencoder. It encodes the input. This encoded input is then decoded to get the output [Zhang *et al.*, (2017)], which is the reconstruction of the input. This is given in Eq. (5) and Eq. (6):

$$h = a_1(W_1 x + b_1) \quad (5)$$

$$x' = a_2(W_2 h + b_2) \quad (6)$$

where k is the number of nodes in the hidden layer and is usually less than the input layer, $W_1 \epsilon R^{kxD}$ and $W_2 \in R^{Dxk}$ are the weight matrices, $b1 \epsilon R^D$ and $b2 \epsilon R^D$ are the bias vectors and a_1, a_2 are non-linear activation functions like Sigmoid, Identity etc. The Autoencoders are trained by back propagating the squared loss [Wu *et al.* (2016)]; [Goodfellow *et al.*, (2016)] with the aim to minimize the reconstruction loss as shown in Eq. (7):

$$\mathcal{L}(x, x') = \|x - x'\| = \left\|x - a_2\left(W_2\left(a_1(W_1 x + b_1)\right) + b_2\right)\right\| \quad (7)$$

2.1.1 Types of autoencoders

Autoencoders are basically classified as under complete AEs, where the size of hidden layer is less than the input size and over complete, where the size of hidden layer can be greater than the input size [Goodfellow *et al.*, (2016)]. Commonly used Autoencoders for recommender systems are Sparse Autoencoders and Denoising Autoencoders. When the network has many layers, the autoencoder's learning process may become very slow, give poor results and lead to identity function. This can be avoided with Sparse AEs which are autoencoders with a sparsity penalty Ω(h) on the hidden layer that can minimize reconstruction error. Denoising autoencoders [Vincent *et al.*, (2010)] are a type of autoencoders where a fraction of input is corrupted and the network learns to denoise the final output. With this the network does not learn to be merely an identity function but also to remove the corruption from the input. This makes the network to learn useful properties which can be helpful for predictions. The loss function for Denoising AE is:

$$\mathcal{L}(x, \widetilde{x'}) \quad (8)$$

where $\widetilde{x'}$ is the corrupted input. For adding noise, the methods used are: masking noise, Gaussian noise and Salt-and-Pepper noise. When a masking noise is used, a percentage of input is set to zero. The de-noising autoencoder now can be trained to predict the missing values.

3. Dropout

Dropout randomly selects a set of neurons and does not consider it for the training phase, i.e., temporarily removes it from the network (its incoming and outgoing connections). Here the probability of a node to be considered or not is a hyper parameter say, p. At every training stage, every node will have a probability (1-p) of dropout or probability p to be considered in the network. Dropout nodes will not be considered for forward pass and hence the weight updates will not be applied to the node during back propagation. When dropout is added to hidden layers, it regularizes the network [Srivastava *et al.*, (2014)]. Dropout added to input layer adds noise to the input and hence can be used with autoencoder to reconstruct the input without noise.

4. Related Work

Deep learning has been used for recommender systems successfully in recent years. Hybrid CF based Autoencoders [Zhang *et al.*, (2017)] utilizes side information of users and items for rating and ranking predictions. The side information addresses the cold start problem. Autorec [Sedhain *et al.*, (2015)] model utilizes the user ratings with Autoencoder to implement user based and item based CF. A hybrid model proposed in [Strub *et al.*, 2016] implements Denoising Autoencoders with and without side information of users and items for improving the performance of hybrid RS. Integrated model [Dong *et al*, (2017)]; [Li *et al.*, (2015)] use AE with additional information for generating the latent representations and use matrix factorization for rating predictions. Most of the implementations have used L_1, L_2 regularizers for reducing the over fitting of the model. Dropout can be used for regularizing the model. Dropout has been proposed to prevent over fitting of the model [Kuchaiev, (2017)]; [Liang *et al.*, (2015)] in the coding layer. DropoutNet [Volkovs *et al.*, (2017)] addresses the cold start problem of RSs and uses dropouts.

5. Methodology

We propose a hybrid model that uses Denoising Autoencoder (DAE) with dropout for predicting user ratings. Dropout is used at input layer with 0.2 dropout probability for adding noise to the input layer and also at the hidden layers for regularization with 0.5 dropout probability. Additional features are used to improve the performance of the model and address the cold start problem. Additional information is useful whenever there are less ratings available and also provides more information about users/items that can be helpful for the learning process. The rating matrix is considered to be R^{mxn} with m users and n items. We use two approaches:

DeepAutoEnCF-U: input to this approach is (r^u, c^u)

DeepAutoEnCF-I: input to this approach is (r^i, c^i)

where r^u, r^i are rating vectors for user u (1,..m) and item i (1,..n) respectively and c^u/c^i is user's/item's additional feature vector. The loss function for DeepAutoEnCF-U is:

$$\mathcal{L} = \left\| r^u - g(h(r^{u'}, c; W), p \right\|^2 \quad (9)$$

The loss function for DeepAutoEnCF-I is:

$$\mathcal{L} = \left\| r^i - g(h(r^{i'}, c; W), p \right\|^2 \quad (10)$$

where $r^{u'}$ and $r^{i'}$ are corrupted inputs and p is the dropout probability at the hidden layer. For user based approach, DeepAutoEnCF-U, additional information used is user's age, gender and movie genre weightage. This unique additional feature, movie genre weightage, calculates the weightage given by users to every movie genre provided in the dataset. It is calculated using the ratings given by users to movies belonging to a particular genre. For every user, top five genres are set to 1 and others a set to zero. For item based approach, DeepAutoEnCF-I, the additional features used are the genres to which a particular movie belongs.

The proposed Autoencoder uses two encoder and two decoder layers. Root Mean Squared Error (RMSE) is back propagated with the aim to minimize the losses.

6. Experimental Results

6.1 Dataset

Several datasets have been made available by GroupLens Research. Two datasets provided by GroupLens are used for evaluating the results of the proposed model, MovieLens-100K and MovieLens-1M [Harper and Konstan, (2015)]. MovieLens 100K dataset provides 100,000 ratings on 1682 movies by 943 users. MovieLens-1M dataset provides ratings on 3952 movies by 6040 users. In both the datasets, each user has rated at least 20 movies.

6.2 Results

The evaluation is done by splitting the datasets into 80/10/10 and 60/20/20 train/validation/test sets. For both the datasets, user additional features are age and gender. For MovieLens-1M dataset the model also uses movie genre weightage feature. Item additional features used are genres of the movies. After experimenting for various values of dropout probability for input layer and hidden layers, it has been set to 0.2 for input layer for adding noise to the input data and to 0.5 for all hidden layers. The size of first encoder is set to 512 and that of second encoder to 256 for MovieLens-1M dataset. Similarly, for MovieLens-100K dataset the sizes of first and second encoders are 135 and 65 respectively. Rectified Linear Units (ReLU) is used as the activation function for all the layers and Adam's optimizer for optimization. Batch Normalization is used for normalizing the inputs of activation functions. RMSE is used for evaluating the rating predictions of both the dataset. Table 1 shows comparison of RMSE over test set for MovieLens-100K dataset for the proposed model with traditional used-based, item-based collaborative filtering techniques and Singular Value Decomposition (SVD). Table 2 shows this comparison for MovieLens-1M dataset.

Table 1. Comparison of Traditional and Proposed Model with RMSE for MovieLens-100K

Algorithm	**MovieLens-100K (80/10/10 split)**	**MovieLens-100K (60/20/20 split)**
User Based CF	1.01	1.20
Item Based CF	1.02	1.23
SVD	0.95	0.96
DeepAutoEnCF-U	**0.911**	**0.901**
DeepAutoEnCF-I	**0.791**	**0.826**

Table 2. Comparison of Traditional and Proposed Model with RMSE for MovieLens-1M dataset (60/20/20 dataset split)

Algorithm	**MovieLens-1M (80/10/10 split)**	**MovieLens-1M (60/20/20 split)**
User Based CF	0.97	0.98
Item Based CF	0.94	1.05
Singular Value Decomposition	0.86	0.89
DeepAutoEnCF-U without features	**0.626**	**0.601**
DeepAutoEnCF-U with features: age and gender	**0.589**	**0.598**
DeepAutoEnCF-U with features: age, gender, movie genre weightage	**0.599**	**0.607**
DeepAutoEnCF-I without features	**0.622**	**0.6196**
DeepAutoEnCF-I with features: movie genres	**0.581**	**0.6069**

Fig. 1 shows the training loss against the validation loss of DeepAutoEnCF-U for MovieLens-1M dataset without any features. Fig. 2 and Fig. 3 show the training loss against the validation loss of DeepAutoEnCF-U on MovieLens-1M dataset with age, gender features and age, gender movie and genre weightage feature respectively. The model with age, gender and movie genre weightage feature performs better than the model without features and with only age and gender feature. Though the RMSE of this model is a bit more than the model with only age and gender feature, the performance graph shows improvement in learning and hence the model can give better performance if we generalize it.

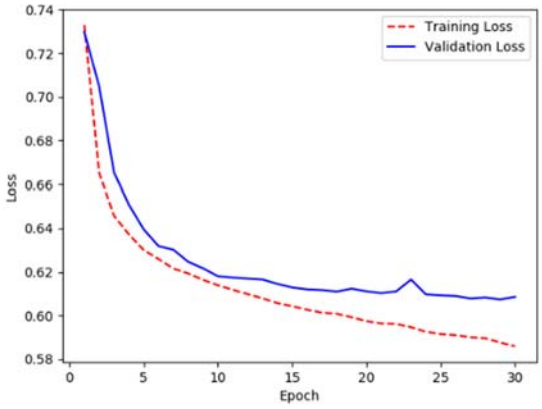

Fig 1. DeepAutoEnCF-U Training Vs. Validation Loss for MovieLens-1M dataset (80/10/10 split) Without Features

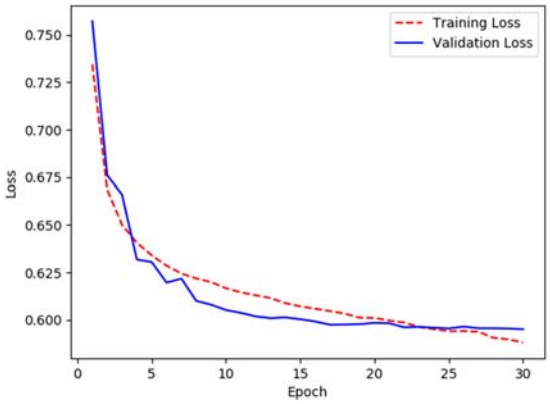

Fig 2. DeepAutoEnCF-U Training Vs. Validation Loss for MovieLens-1M dataset (80/10/10 split) with Age and Gender Features

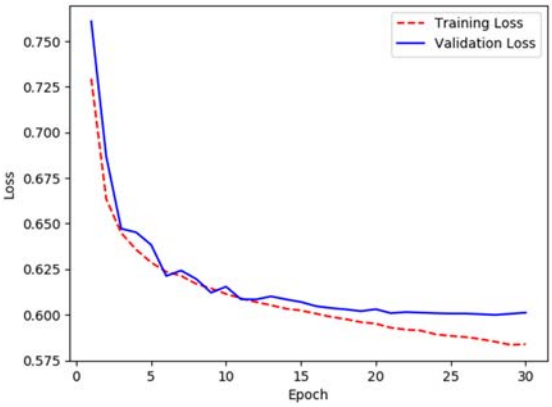

Fig 3. DeepAutoEnCF-U Training Vs. Validation Loss for MovieLens-1M dataset (80/10/10 split) with Age, Gender and Movie Genre Weightage Features

Fig. 4 shows performance of DeepAutoEnCF-U on MovieLens-1M dataset for 60/20/20 split with features. The model overfits for this split with features. It shows similar performance with and without features for this split.

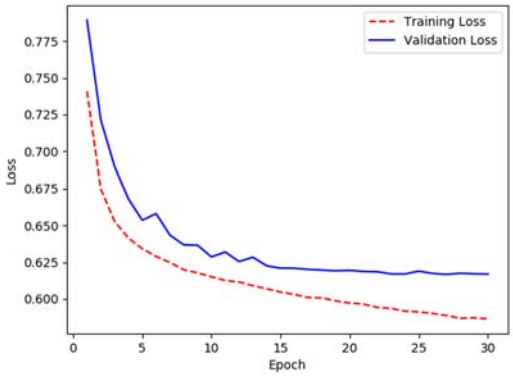

Fig 4. DeepAutoEnCF-U Training Vs. Validation Loss for MovieLens-1M dataset (60/20/20 split) with Age, Gender and Movie Genre Weightage Features

Fig. 5 shows the training loss against the validation loss of DeepAutoEnCF-I for MovieLens-1M dataset without any features. Fig. 6 shows the training loss against the validation loss of DeepAutoEnCF-I on MovieLens-1M dataset with movie genre feature. Here also the model performs better with features which can perform better when the model is generalized.

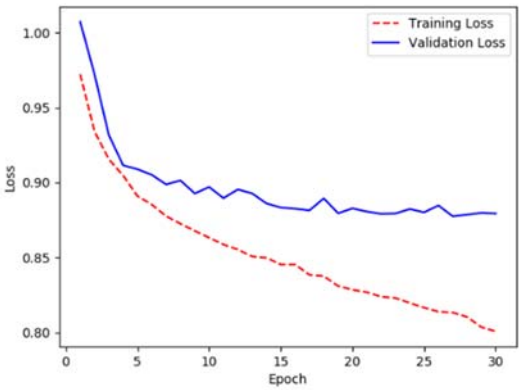

Fig 5. DeepAutoEnCF-I Training Vs. Validation Loss for MovieLens-1M dataset (80/10/10 split) without Features

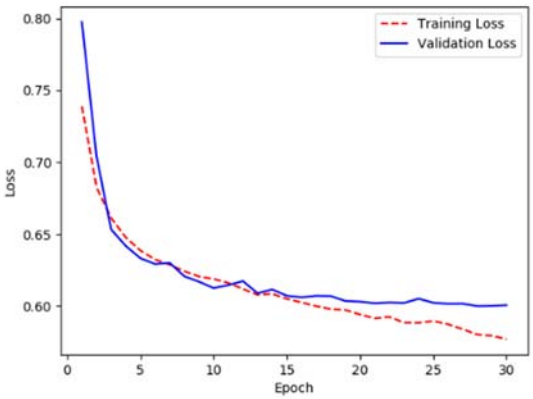

Fig 6. DeepAutoEnCF-I Training Vs. Validation Loss for MovieLens-1M Dataset (80/10/10 split) with Movie Genre Feature

Fig. 7 shows performance of DeepAutoEnCF-I for MovieLen-1M dataset for 60/20/20 dataset split. For item based approach also the model shows either overfit or unpresentable performance. For both the approaches the performance (RMSE) and the learning improves when features are added to the input with 90/10/10 dataset split.

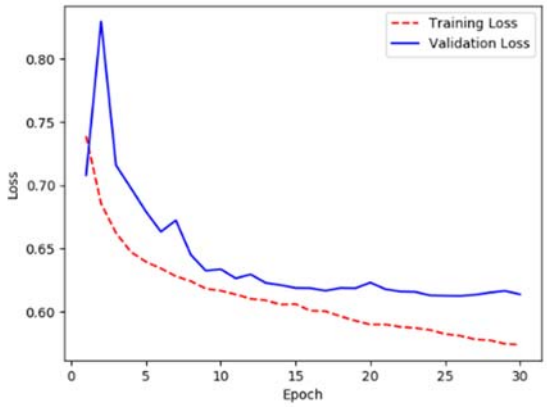

Fig 7. DeepAutoEnCF-I Training Vs. Validation Loss for MovieLens-1M Dataset (60/20/20 split) with Movie Genre Feature

7. Conclusions

For both the datasets, the proposed system outperforms the traditional collaborative filtering algorithms. For MovieLens 100K dataset DeepAutoEnCF-I showed better performance than DeepAutoEnCF-U. For MovieLens-1M dataset both the approaches, user based and item based, show improvement in performance when features are added to the input. The performance of DeepAutoEnCF-I improved more than DeepAutoEnCF-U as compared to their performance without features. Whenever the data is spare, adding new features supports the learning process and improves the performance. Denoising Autoencoders with dropout improve the performance of recommender systems for predicting ratings. New features can be thought and added to the input further for improving the performance of recommender systems with deep learning through denoising autoencoders.

References

[1] Betru B. T., Onana C. A. and Batchakui B. (2017), "Deep Learning Methods on Recommender System: A Survey of State-of-the-Art", International Journal of Computer Applications, Volume 162 – No – 10, March 2017.
[2] Dong X.,et al. (2017), "A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems", Thirty first AAAI Conference on Artificial Intelligence.
[3] Ekstrand M. D., Riedl J. T. and Konstan J. A. (2018), "Collaborative Filtering Recommender Systems", Foundations and Trends in Human–Computer Interaction Vol. 4, No. 2, 81–173.
[4] Goodfellow I., Bengio Y. and Courville A. (2016), "Deep Learning", MIT Press.
[5] Harper M. and Kostan J. A. (2015), "The MovieLens Datasets: History and Context", ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19, 19 pages. DOI = http://dx.doi.org/10.1145/2827872
[6] Isinkaye F. O., Folajimi Y. O. and Ojokoh B. A. (2015), "Recommendation systems: Principles, methods and evaluation", Egyptian Informatics Journal 16, 261-273.
[7] Koren Y. (2009), "Matrix Factorization Techniques for Recommender Systems", IEEE Computer Society.
[8] Kuchaiev O. and Ginsburg B. (2017), "Training Deep AutoEncoders for Collaborative Filtering", arXiv.
[9] Li S., Kawale J. and Fu Y. (2015), "Deep Collaborative Filtering via Marginalized Denoising Auto-encoder", CIKM, ACM.
[10] Liang J. and Liu R. (2015), "Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network", 8th International Congress on Image and Signal Processing (CISP).
[11] Sarwar B., et al. (2001), "Item-Based Collaborative Filtering Recommendation Algorithms", WWW10, ACM.
[12] Sedhain S., et al. (2015), "AutoRec: AutoEncoders meet collaborative filtering", WWW, ACM.
[13] Srivastava N., et al. (2014), "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Journal of Machine Learning Research 15, 1929-1958.
[14] Strub F., Gaudel R. and Jeremie M. (2016), "Hybrid Recommender System Based on Autoencoders", Workshop on Deep Learning for Recommender Systems, Boston US, ACM.
[15] Su X. and Khoshgoftaar T. M. (2009), "A Survey of Collaborative Filtering Techniques", Advances in Artificial Intelligence.
[16] Vincent P., et al. (2010), "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion", Journal of Machine Learning Research 11, 3371-3408.
[17] Volkovs M., Yu G. and Poutanen T. (2017), "DropoutNet: Addressing Cold Start in Recommender Systems", NIPS.
[18] Wu Y., et al. (2016), "Collaborative Denoising Auto-Encoders for Top-N Recommender Systems", WSDM, ACM.
[19] Zhang S., et al. (2017), "Hybrid Collaborative Recommendation via Semi-AutoEncoder", 4th International Conference, ICONIP.
[20] Zhang S., Yao L. and Sun A. (2017), "Deep Learning Based Recommender System: A Survey and New Perspectives", ACM J. Comput. Cult. Herit., Vol. 1, No. 1, Article 35.