Wide & Deep Learning in Job Recommendation: An Empirical Study

Shaoyun Shi, Min Zhang(✉), Hongyu Lu, Yiqun Liu, and Shaopin Ma

Tsinghua National Laboratory for Information Science and Technology,
Department of Computer Science and Technology,
Tsinghua University, Beijing 100084, China
{shisy17,luhy16}@mails.tsinghua.edu.cn,
{z-m,yiqunliu,msp}@tsinghua.edu.cn

Abstract. Recommender systems have become more and more popular in recent years. Collaborative Filtering and Content-Based methods are widely used for a long time. Recently, some researchers introduced deep learning algorithms into recommender system. In this paper, we try to answer some questions about a novel recommender model, Wide & Deep Learning. Firstly, how should we select and feed in features? Secondly, how does Wide & Deep Learning work? Thirdly, how to joint-train the two parts of the network? Finally, how to conduct online training with new data? For all of these, we focus on the job recommendation task, which often suffers from the cold-start problem. The experiments give us the answers of these questions.

Keywords: Recommender system · Wide & Deep Learning · Job recommendation

1 Introduction

With the continuous development of the Internet, information explosion has become a great challenge that people are faced with [1]. How to get the information we need is a big problem in such a situation. As a result, recommender systems are designed to help deal with this problem. Collaborative Filtering and Content-Based methods are widely used for a long time and many studies are based on them. On the other hand, deep learning methods [2] have achieved remarkable results in many fields like image recognition [3] and natural language processing [4], which have also been introduced into recommender systems recently to solve some problems, such as the cold-start problem.

Cold-start is a serious problem in recommendation [5]. There are new items and users coming to the system every day and we do not have any historical information of them. For example, the job recommendation task suffers from the item cold-start problem. There are new jobs published on the website every

This work is supported by Natural Science Foundation of China (Grant No. 61532011, 61672311) and National Key Basic Research Program (2015CB358700).

W.-K. Sung et al. (Eds.): AIRS 2017, LNCS 10648, pp. 112–124, 2017.
https://doi.org/10.1007/978-3-319-70145-5_9

day, which makes it necessary to update the recommendation model. Although traditional Collaborative Filtering methods get remarkable performance, they are weak on solving the cold-start problem because it is difficult for them to update model when fresh users and items come. To better take advantage of users' and items' profile and other external information, deep learning is taken into consideration. Most combinations are implemented by combining deep concepts with traditional methods, such as optimizing the loss of user/item vectors in matrix decompositions by far.

However, there are also new models proposed, such as Wide & Deep Learning [6], which is a novel Content-Based method proposed by Google. Researchers have shown that deep learning has a strong ability to generalize, besides, traditional linear models have a great ability of "memorizing". Both of the two properties are essential in recommendations. Based on this idea, Google proposed Wide & Deep Learning, combining deep neural network with the linear model. It can be regarded as a Content-Based method because the input features are mainly about users' and items' profile and the ratings or other kinds of interaction history are used to supervise the model training.

However, there are still many uncertain questions in applying Wide & Deep. Different kinds of machine learning methods have different abilities to process different kinds of features and data. So what kinds of features should we feed into the Wide & Deep network? Is feature selection necessary? And questions about how Wide & Deep works, such as which part is more important, still have no answer. What's more, due to the phenomenon that the two parts' learning speed is not the same, we need to ensure that the two parts are both well-trained by effective joint-training strategy. Besides, in real-world recommender systems, because of the new data coming every day, it is important to update the model efficiently. We did experiments on all of these questions and got some insights on how to use the Wide & Deep neural network.

The contributions of our work are listed as following:

- We summarized several questions about Wide & Deep which have not been answered by far, while they are important in applying this algorithm.
- Several analysis methods, e.g.: Layer-wise Relevance Propagation (LRP), and Experiments are conducted in job recommendation task to seek the answers to the questions.
- We applied Wide & Deep Learning to job recommendation and achieved good performance.

2 Related Work

2.1 Traditional Methods

The two traditional recommendation techniques are Content-Based and Collaborative Filtering.

Many outstanding Collaborative Filtering methods are based on matrix factorization, such as WRMF [7], ExpoMF [8], MMMF [9] and BPR-MF [10]. Given

an interaction matrix $R = (R_{ij})_{m \times n} \in \mathfrak{R}_{+}^{m \times n}$ of m users and n items, the goal is to get user matrix $U_{m \times k}$ and item matrix $V_{k \times n}$. $R = UV$, where k is the dimension of latent vector. Each row of U represents a user vector u_i and each column of V represents an item vector v_j. We can use u_i and v_j's dot product as the rating prediction about how user i likes item j. But these methods can not deal with users or items with no historical information and it is really hard to add new users and items unless you re-train the model.

On the other hand, Content-Based methods make recommendation mainly based on users' and items' profile and contents [11]. The Content-Based methods can handle the item cold-start problem in recommendation because they directly extract information from the items' content and are independent of history information. Besides, Content-Based methods are often regarded as having a better explanation for recommendation, which is also very important in the recommender system.

2.2 Deep Learning in Recommendation

Traditional Content-Based methods are weak in processing complex text, image or audio information. Due to the remarkable performance deep learning gets in other fields recent years, many researchers are trying to introduce deep learning in recommender systems. Some are trying to strengthen the ability of Content-Based methods with deep learning. For example, Van used CNN to extract latent vectors from music audio and achieved excellent performance compared to principal components analysis (PCA) based method [12].

There are also some researchers who want to combine the deep learning with Collaborative Filtering. One way is to optimize the latent vector in matrix factorizing based methods. For example, Marginalized Denoising Auto-encoder based Collaborative Filtering (mDA-CF) [13] combines probabilistic matrix factorization with marginalized denoising stacked auto-encoders.

But these models still can not apply deep learning independently with traditional methods. Some works tried to do Collaborative Filtering alternatively by deep neural network [14–16]. The Neural Collaborative Filtering (NCF) model proposed by He et al. achieved significant improvements over many state-of-the-art methods. Besides, Google recently proposed a novel recommender model, Wide & Deep Learning, a state-of-the-art Content-Based method we would like to focus on in this paper.

3 Wide & Deep Learning

The Wide & Deep Learning was proposed by Google, whose motivation was to combine the advantages of deep neural network and linear model. Deep neural network has a strong ability to process sparse features like text and can extract dense embeddings which contain much information and are easier to use. It is considered to have a strong ability to generalize. But sometimes the deep model may over-generalize and recommend some less relevant items to the users, which is undesirable in the recommender system. One way to solve this problem is to

combine the deep neural network with the linear model, which is considered to have a strong ability to memorize. It can learn a direct relationship between input features and output targets.

Google combines the deep neural network and linear model with logistic regression, shown in Fig. 1, where y_{ui} and $\hat{y}_{ui}$ denote the real and predicted label respectively.

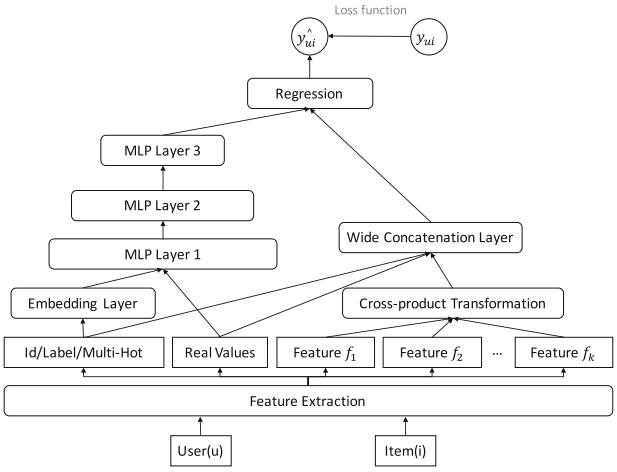

Fig. 1. The structure of Wide & Deep Learning

To formalize the prediction, let Y denote the class label and $\mathbf{x}$ denote the original features, then we have:

$$P(Y = 1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^{\top}[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^{\top} a^{(l_f)} + b) \tag{1}$$

where $\sigma(\cdot)$ is the sigmoid function, $\phi(\mathbf{x})$ are the cross product transformations of the original features, and b is the bias. $\mathbf{w}_{wide}$ is the vector of all wide model weights, and $\mathbf{w}_{deep}$ is the weights applied on the final activations $a^{(l_f)}$.

The cross product transformations on the linear part are based on experience added artificially. It represents features' simultaneous appearance. For example, we have a 2-dimension one-hot vector for one's gender, and a 2-dimension one-hot vector for one's country (if Germany or not). Then we can generate a 4-dimension one-hot vector, which represents if one is a German man, woman or other countries' man or woman. We can define the cross-product transformation as follows:

$$\phi_k(\mathbf{x}) = \prod_{i=1}^{d} x_i^{c_{ki}} \quad c_{ki} \in \{0, 1\} \tag{2}$$

where ϕ_k represents the k-th transformation, c_{ki} is a boolean variable denoting whether ϕ_k is related to the i-th feature x_i.

We implemented the model with Tensorflow[1]. But there are still many questions on the model and how to take advantage of it:

1. Is feature selection necessary in the model and how to do it?
2. How do the deep neural network and linear model work with each other?
3. How can we avoid over-fitting of one part but at same time under-fitting of another part?
4. In real-world recommender systems, how can we conduct an online training with new data efficiently?

We will give discussions and experiments results on all questions above in following sections.

4 Empirical Study on How Wide & Deep Learning Works

4.1 Job Recommendation

Our experiments were conducted on job recommendation task, which can be regarded as a special task different from some regular recommendation task because it suffers from the item cold-start problem. Jobs published on websites every day are all new items and have no historical information. As only as a job is taken by enough applicants, it is no longer available. New jobs are recommended only according to their profile and users' historical information.

The dataset is from RecSys Challenge 2017[2]. The challenge is comprises into two parts, offline and online test. The size of training data and target data is shown in Table 1.

Table 1. The size of data in RecSys Challenge 2017

	Training			Target	
	Users	Items	Interactions	Users	Items
Offline test	1,497,020	1,306,054	322,766,002	74,840	46,559(41047[a])
Online test	981,673	1,037,282	92,949,362	48,167	1k–15k(all[a])

[a] Number of cold-start items

The goal is to recommend jobs to users. There is a specific algorithm defined by the organizers of the challenge to calculate the score. U denotes the target users and I denotes the target items. I_u denotes the items recommended to user $u \in U$ and U_i denotes the users item $i \in I$ is recommended to. Then we have:

$$Score = \sum_{u \in U} \sum_{i \in I_u} userSuccess(i, u) + \sum_{i \in I} itemSuccess(i) \tag{3}$$

[1] https://www.tensorflow.org.
[2] http://2017.recsyschallenge.com/.

in which

$$\begin{aligned} &userSuccess(i,u) = \\ &\quad (1 * click(i,u) + 5 * reply\_or\_bookmark(i,u) + 20 * recruiter(i,u) \\ &\quad - 10 * delete\_only(i,u)) * (premium(u) + 1) \\ &itemSuccess(i) = \\ &\quad (|\{userSuccess(i,u) | userSuccess(i,u) > 0, u \in U_i\}| > 0) \\ &\quad * 25 * (paid(i) + 1) \end{aligned} \tag{4}$$

The size of users and items are too big for matrix factorization based methods. Collaborative Filtering methods do not work on the cold-start problem. Besides, the recommendation can also be considered as a classification problem, which all user-item pairs have a binary class label, like or dislike. As a result, Wide & Deep Learning, a state-of-the-art Content-Based method, is suitable.

4.2 Feature Selection for Wide & Deep

The deep neural networks are usually thought to have the ability to extract and select features automatically. For example, they can capture the lines and edges to understand the whole image. All we need to feed into them is the raw presentation of the images. However, Wide & Deep Learning is not a traditional end-to-end use of neural network to process image or text information but a non-linear classifier, whose input still includes features extracted by users of the model. We wonder if feature selection is still necessary.

Many item features and user features were extracted. Especially, though we don't have the history of some items, we can take advantage of user history. We extract many pairwise features, e.g.: which item tag or other attributes did user prefer in the history. Most of them are numeric features which usually are considered really important in the classifiers. Categorical features like country and discipline first pass through an embedding layer and then concatenate with other features as the input of hidden layers in deep part. Cross features we generate are mainly cross transformations between discipline and industry.

We compared the performance between the models with and without feature selection. To conduct feature selection, we calculated the Pearson correlation between the label and all the numeric features, and remove all the features whose Pearson correlation are lower than 0.2. Some experimental results on feature selection are shown in Table 2.

Parameters like learning rate, number and size of hidden layers are all well tuned separately in our models. We use Adam [17] as the optimizer and Cross-Entropy [18] as the loss function. Dropout has been tried but did not show remarkable effect. Early stop is conducted in the training process.

It can been seen that the performance after feature selection is much better than using all features, which show that feature selection is necessary to Wide & Deep Learning. Some features may be harmful to the network. Besides, historical pair features we extract are really important. They are necessary to solve the item

Table 2. Experiments on feature selection

	All features	No Historical[a]	Feature selection
Score	60	903	26,855

[a] The difference with "Feature Selection" is that there are no historical pair features which capture how the user prefers the item's tags or other attributes in the history.

cold-start problem for that they match the item attributes with users' history. Experiments in the following sections are all done with feature selection.

4.3 Roles of Wide and Deep Components in Learning

The two components of Wide & Deep are really different. They should play different roles in the model. Which part is more important and if they have different understandings to the input features are still unknown. Deep neural networks are always been regarded as black boxes, but there are new technologies can help understand them.

Layer-wise Relevance Propagation (LRP) is a state-of-the-art technique used in the field of image processing [19,20] and natural language processing [21,22]. It has been used to visualize what the neural network has learned and what it thinks is important. It back-propagates the LRP relevance from the higher layers to the lower layers according to the weight parameters to find out which part contributes most to the final result. Here we would like to introduce LRP to Wide & Deep to help us understand the network and better take advantage of it.

Feed-forward propagation in our Wide & Deep network can be defined as

$$x_j^{l+1} = \sigma(\sum_i x_i^l w_{ij}^{l,l+1} + b_j^{l+1}), \quad e.g. \quad \sigma(z) = relu(z) = max(0, z) \tag{5}$$

where j is the index of a particular neuron at layer $l+1$, $w_{ij}^{l,l+1}$ and b_j^{l+1} are elements of weight matrix and bias from layer l to layer $l+1$. $\sigma(\cdot)$ is the activation function. Make R_i^l be the relevance score of i-th neuron at layer l, then we have

$$R_i^l = \sum_j \frac{z_{ij}}{\sum_k z_{kj} + \epsilon sign(\sum_k z_{kj})} R_j^{l+1} \quad with \quad z_{ij} = x_i^l w_{ij}^{l,l+1} \tag{6}$$

where ϵ is a small positive number to make the computation more stable and have better numerical properties.

We focus on two questions: Which part is more important? Does the wide part have better ability of memorizing sparse features' occurrence?

- The total relevance scores of deep part and wide part were first calculated, as shown in Table 3.

Table 3. Overall relevance score of deep part and wide part

	Deep	Wide
LRP Relevance	0.748	0.252

It can be seen that the deep neural network plays a much more important role in the Wide & Deep in this problem. It is reasonable because deep neural networks are supposed to have a better descriptive ability and are able to capture the relationship between features. To verify the importance of deep model, we also change the position of numeric features (which are the most important features in the network). Experiment results are shown in Table 4. A score is fed back for each upload. We show the average of 5 uploads.

Table 4. Experiments on numeric features' positions. ("Deep-Only"/"Both" is significantly better than "Wide-Only" with p-value $9e-5/3e-4$. But the difference between "Deep-Only" and "Both" is not significant in terms of p-value 0.15.)

	Wide-Only	Deep-Only	Both
Offline score	24,163	25,873	25,568

The result shows that if the deep part lacks important features, the performance of the entire model will significantly drop. If these features are fed into both wide and deep part, the performance is almost the same as not feeding into the wide part. This also verifies that deep part is the main part.

- Detailedly, the LRP relevance scores of features in two parts were calculated. The results are shown in Fig. 2.

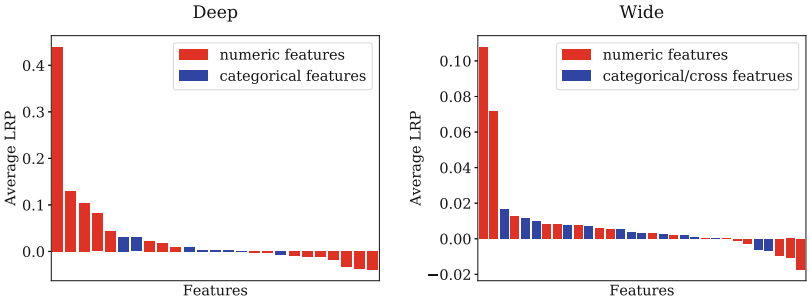

Fig. 2. LRP relevance of some features in two parts.

Although the feature with the highest relevance is a numeric feature in both parts, categorical and cross features have greater impacts in wide part than

in the deep part. Since they are one-hot vectors which indicate the occurrence of properties, it shows that wide linear part memorizes the direct relationship between the occurrence and final label.

4.4 Comparative Study on Training Strategies

Different models have different learning speed on the same data. We want to investigate if it is a good way to combine the two kinds of model and train together. One of the concerns is that when one of the two parts have over-fitted, another part is still under-fitting. To avoid this, we tried different strategies of joint-training, including training deep part after training together or training together after training the two single parts. In each training period, we stop training when the cost on validation set doesn't get lower for 10 epochs. The average offline upload score and training epochs of 3 repeated experiments are shown in Table 5.

Table 5. Offline upload score of different joint-training strategy

Strategy[a]	Offline score	Epochs
Wide→Deep→Collaboratively	22,938	77
Deep→Wide→Collaboratively	25,354	50
Collaboratively→Deep	25,513	46
Collaboratively→Wide	25,727	34
Collaboratively	**25,786**	**22**

[a] When training a single part, we keep another part unchangeable and optimize the cost of the whole model. Otherwise, collaboratively training update the parameters of two parts at the same time.

It can be seen that the scores are almost the same and training collaboratively is slightly better than other strategies. Training together after training the wide part and deep part has a worse performance because it is not very stable. But simply training collaboratively need the least training epochs and can be regarded as the fastest and the most efficient way.

The reason we think is that the two parts may work together and help each other while training and predicting. We record the cost on validation set of two separate parts during training collaboratively, as shown in Fig. 3.

Although deep part achieves a stable status, the cost of wide part is still fluctuating, the entire model has lower cost than both of them and are more stable. Interestingly, in epochs 8 and 9, the costs of wide part and deep part fluctuate at different directions, but the cost of entire model drops. The Wide & Deep Learning is different from ensemble learning. Joint-training in Wide & Deep makes the two parts help and learn from each other during training collaboratively.

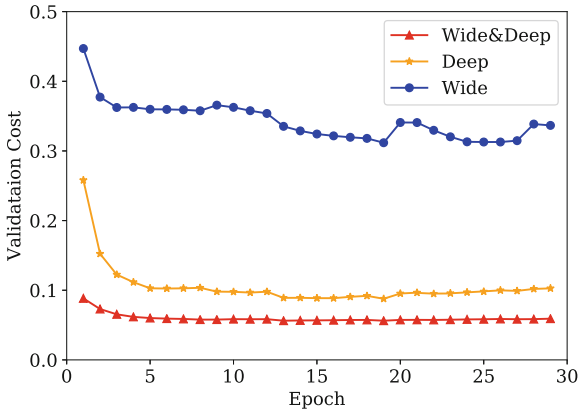

Fig. 3. Cost on validation set of two parts (Wide/Deep) while training.

4.5 Online Training

In real-world recommender system, new data is coming every day. Preferences of users and attributes of items may change over time and influence each other [23]. How to conduct an online training and update the model regularly is important so that it can keep a good performance.

In this job recommendation task, data provided in the online test is different from that provided in the offline test. The new data contains two kinds of information: new training data and everyday interaction feedback. The most straightforward way is to re-train the recommendation model every day. However, it may cost too much time and computing resources. Another way is to update the model in an incremental way, which means we can load the previous model and continue training with fresh data. It is necessary to choose an online training strategy in the challenge. Since that only one strategy can be upload each day and the number of target items differs between days, it is unfair to compare the scores calculated from feedbacks whose amount is also changeable. To conduct a fair comparison, we generate results on one day's targets with enough feedback and measure the performances on our own. The precision (positive interaction number/impression number) of different model update strategies is calculated. The results are shown in Table 6.

Table 6. Performance of different model update strategies

Model	Online-only	Fine-tuning[a]	Re-train
Precision	0.174	0.311	0.384

[a] Load original model with best performance in the offline test and train with new data.

It makes sense that the re-trained model gets the best performance, but actually, the time of training is almost two times longer than the fine-tuning

model. In this task, it is acceptable. However in the real-world problem with larger scale of data, re-training the model every time is not necessary and fine-tuning is a more efficient strategy which can provide acceptable performance.

4.6 Overall Recommendation Performance

The Wide & Deep achieves excellent performance. Comparison between Wide & Deep Learning and other models is shown in Table 7. All models are tuned at our best. Due to the different amounts of targets in online test every day, we calculate the average score per item each method gets during the days to show an overall performance.

Table 7. The performance of different models

	Offline score	Online score/Item
Item-neighbour	12,438	1.0180
Historical Enhancement	20,450	0.9541
XGBoost	14,628	—
Logistic Regression (Wide)	24,168	—
Neural Network (Deep)	25,539	—
Wide & Deep	**26,855**	**1.0210**

The performance of Wide & Deep Learning is better than other models. Especially, the Wide & Deep achieves better performance than both the wide model and the deep model, which shows that the combination is reasonable.

5 Conclusion

In this paper, our aim is not to propose a new model, but to conduct an empirical insight study of a novel model, Wide & Deep Learning, in real scenario with large scale online data. We want to better understand and take advantage of it. Although the experiments were done in only one field, job recommendation, the datasets in offline and online test are different. The offline dataset includes some instances artificially added by organizers and the online test has a more real evaluation. The two tests can be regarded as two separated problems. Our contributions can be concluded as follows:

1. We found that feature selection is still necessary in this model. Irrelevant features are harmful to the model's performance.
2. We used LRP to analyze the network and results shows that deep part takes an important role in the network, important features like numeric features must not be absent in the deep neural network.

3. We conclude that joint-training the wide and deep parts collaboratively is the most efficient and effective way.
4. We discussed the strategies to conduct an online training. Re-training the model with whole data set is best with enough time and computing resources. Loading the old model and continue to train is more efficient cost of a little lower performance.

As a state-of-the-art Content-Based method, Wide & Deep combines the generalizing ability of deep neural network and memorizing ability of the wide linear model. But the structure currently lacks the ability to process text or image information. How to import CNN or other kinds of the network to the model and apply it to other datasets and problems is a good question and will be included in our future work.

References

1. Sweeney, L.: Confidentiality, disclosure, and data access: theory and practical applications for statistical agencies. Inform. Explosion, 43–74 (2001)
2. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
3. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
4. Manning, C., Socher, R., Fang, G.G., et al.: CS224n: natural language processing with deep learning1 (2017)
5. Son, L.H.: Dealing with the new user cold-start problem in recommender systems: a comparative review. Inform. Syst. **58**, 87–104 (2016)
6. Cheng, H.T., Koc, L., Harmsen, J., et al.: Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp. 7–10. ACM (2016)
7. Pan, R., Zhou, Y., Cao, B., et al.: One-class collaborative filtering. In: Eighth IEEE International Conference on Data Mining, ICDM 2008, pp. 502–511. IEEE (2008)
8. Liang, D., Charlin, L., McInerney, J., et al.: Modeling user exposure in recommendation. In: Proceedings of the 25th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, pp. 951–961 (2016)
9. Weimer, M., Karatzoglou, A., Smola, A.: Improving maximum margin matrix factorization. Mach. Learn. **72**(3), 263–276 (2008)
10. Rendle, S., Freudenthaler, C., Gantner, Z., et al.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 452–461. AUAI Press (2009)
11. Lops, P., De Gemmis, M., Semeraro, G.: Content-based recommender systems: state of the art and trends. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.) Recommender Systems Handbook. Springer, Boston (2011). doi:10.1007/978-0-387-85820-3_3
12. Van den Oord, A., Dieleman, S., Schrauwen, B.: Deep content-based music recommendation. In: Advances in Neural Information Processing Systems, pp. 2643–2651 (2013)

13. Li, S., Kawale, J., Fu, Y.: Deep collaborative filtering via marginalized denoising auto-encoder. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management [S.l.], pp. 811–820. ACM (2015)
14. He, X., Liao, L., Zhang, H., et al.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, pp. 173–182 (2017)
15. Zheng, Y., Tang, B., Ding, W., et al.: A neural autoregressive approach to collaborative filtering. arXiv preprint arXiv:1605.09477 (2016)
16. Zheng, Y., Liu, C., Tang, B., et al.: Neural autoregressive Collaborative Filtering for implicit feedback. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems [S.l.], pp. 2–6. ACM, 2016
17. Kingma, D., Adam, B.J.: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Deng, L.Y.: The cross-entropy method: a unified approach to combinatorial optimization. Monte-Carlo Simul. Mach. Learn. (2006)
19. Bach, S., Binder, A., Montavon, G., et al.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS One **10**(7), e0130140 (2015)
20. Binder, A., Bach, S., Montavon, G., Müller, K.-R., Samek, W.: Layer-wise relevance propagation for deep neural network architectures. Information Science and Applications (ICISA) 2016. LNEE, vol. 376, pp. 913–922. Springer, Singapore (2016). doi:10.1007/978-981-10-0557-2_87
21. Ding, Y., Liu, Y., Luan, H., et al.: Visualizing and understanding neural machine translation
22. Arras, L., Montavon, G., Müller, K.R., et al.: Explaining recurrent neural network predictions in sentiment analysis. arXiv preprint arXiv:1706.07206 (2017)
23. Dai, H., Wang, Y., Trivedi, R., et al.: Recurrent coevolutionary feature embedding processes for recommendation. arXiv preprint arXiv:1609.03675 (2016)