ODKRYWANIE WIEDZY I SYSTEMY REKOMENDACYJNE

Jakub Sachajko 179976
Przemysław Rośleń 180150

Article: Neural Collaborative Filtering

link: https://arxiv.org/pdf/1708.05031v2.pdf

Article authors:

Xiangnan He National University of Singapore, Singapore xiangnanhe@gmail.com
Lizi Liao National University of Singapore, Singapore liaolizi.llz@gmail.com
Hanwang Zhang Columbia University USA hanwangzhang@gmail.com
Liqiang Nie Shandong University China nieliqiang@gmail.com
Xia Hu Texas A&M University USA hu@cse.tamu.edu
Tat-Seng Chua National University of Singapore, Singapore dcscts@nus.edu.sg

Etap I – Article analysis

1. Motivations

a) In the era of information explosion, recommender systems play a huge role in mitigating information overload. They are widely used by many services, including E-commerce, online news, and social media sites. The online key to a personalized recommender system is in predicting users' preference for items based on their past interactions (e.g., ratings and clicks), known as collaborative filtering.

   The key problem addressed in this article is the application of deep neural networks to collaborative filtering in recommender systems, specifically focusing on implicit feedback.

b) Among the various collaborative filtering techniques, matrix factorization (MF) is the most popular one, which projects users and items into a shared latent space, using a vector of latent features to represent a user or an item. Thereafter a user's interaction on an item is modelled as the inner product of their latent vectors. Popularized by the Netflix Prize, MF has become the de facto approach to latent factor model-based recommendation.

   Advantages:

- Dimensionality Reduction: Matrix factorization can reduce the dimensionality of data, making it easier to work with and interpret. This can be particularly useful for feature selection or data compression.

- Collaborative Filtering: In recommendation systems, matrix factorization is often used for collaborative filtering, where it helps to discover latent patterns in user-item interactions. This leads to more personalized and accurate recommendations.

Disadvantages:

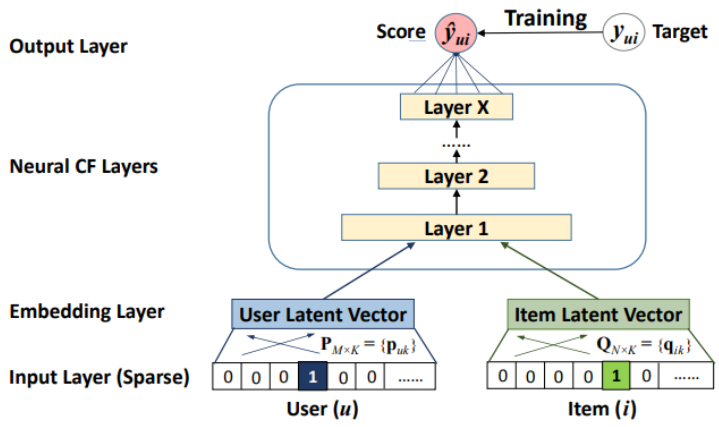
- Cold Start Problem: Matrix factorization-based recommendation systems may struggle with the "cold start" problem, where it's challenging to provide recommendations for new users or items with limited interaction data.

- Scalability Concerns: Large-scale matrix factorization problems can be computationally intensive and require significant computational resources, which may not be feasible for all applications.

c) The authors noticed, that while deep learning has seen great success in areas like speech recognition, computer vision, and natural language processing, its application to recommendation systems has not been as thoroughly explored. Deep learning techniques have shown remarkable success in various fields, such as natural language processing, computer vision, and speech recognition. Extending these methods to recommender systems has the potential to significantly improve the accuracy and effectiveness of personalized recommendations. This is a good reason for them to engage in this subject.

d) The neural network's ability to approximate continuous functions is well-established, and deep neural networks (DNNs) have proven effective in various domains, including computer vision, speech recognition, and text processing. However, there has been limited research on using DNNs for recommendation systems compared to the extensive literature on matrix factorization (MF) methods. While DNNs have been mainly employed to model additional information like item descriptions, audio features in music, and image content, some recent advances have applied DNNs to recommendation tasks and shown promising results.

2. Algorithm



1. Firstly, gathering the data into the matrix that contains interactions between users and items, for example, clicks or ratings. And converting it to two feature vectors that describe the user and the item.

2. Changing the encoding. Converting the data to for example one-hot encoding

3. Transform one-hot encoded features into dense vectors (initialized embedding layer).

4. Combine user and item embeddings, either by concatenating or multiplying them.

5. The user embedding and item embedding are then fed into the neural collaborative filtering layers (neural network architecture).

6. Calculate the final predicted score (ˆyui) for every user-item pair. Predicted ˆyui has to be in the range of [0, 1], which can be easily achieved by using a probabilistic function (e.g., the Logistic function) as the activation function for the output layer φout.

$$\hat{y}_{ui} = f(\mathbf{P}^T\mathbf{v}_u^U, \mathbf{Q}^T\mathbf{v}_i^I|\mathbf{P}, \mathbf{Q}, \Theta_f),$$

$$f(\mathbf{P}^T\mathbf{v}_u^U, \mathbf{Q}^T\mathbf{v}_i^I) = \phi_{out}(\phi_X(...\phi_2(\phi_1(\mathbf{P}^T\mathbf{v}_u^U, \mathbf{Q}^T\mathbf{v}_i^I))...)),$$

7. Define a loss function to measure the difference between the predicted score and the actual interaction value (yui). In the given article regression with squared loss has been used. Additionally, they used wui which is the hyperparameter denoting the weight of the training instance (u, i)

$$L_{sqr} = \sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} w_{ui}(y_{ui} - \hat{y}_{ui})^2$$

8. Optimize the model parameters by minimizing the loss using stochastic gradient descent (SGD) or another optimization method.

b) Constants in the scope of single training:

- u - user vector size
- i - item vector size
- yui - ground truth representing user-item interaction strength for user u and item i
- vu - feature input vector containing information about users
- vi - feature input vector containing information about items
- wui - hyperparameter weight representation in the loss function
- φ1, φ2, …, φout - multi-layer neural network layers used in the training process

Variables:

- ^yui - a predicted value representing user-item interaction strength for user u and item i
- f - multi-layer neural network
- Θf - parameters of the interaction function f (neural network parameters)
- K - dimension of latent factors
- P - user latent factor matrix, depends on the number of users, modified during training
- Q - item latent factor matrix, depends on the number of items, modified during training
- Lsqr - squared loss function
- Y - denotes the set of observed interactions
- Y- − denotes the set of negative instances, which can be all unobserved interactions

c) **Input format:**

- The input includes user-item interaction data. The input layer consists of two feature vectors Vu and Vi that describe user u and item i. The exemplar format of numbers in the vectors mentioned in the article consists of 1 and 0, depending on whether the user interacts with the item. It is also possible to format the float type numbers when it comes to showing the strength of the interaction between user and item, for example, ratings from 1 to 5.

Output format:

- The output includes a predicted score for each user-item pair. The exemplar format mentioned in the article assumes float numbers in the range [0, 1], depending on how strongly the user interacts with the item.

d) The Neural Collaborative Filtering (NCF) algorithm is typically used in the two main phases recommendation process:

- Training Phase: During this phase, the NCF algorithm is trained on available historical data related to user behaviors, such as product ratings or interactions. During the training, the NCF model analyzes information about both users and products to learn the dependencies between these elements. The primary objectives are to gather information on users' preferences and product features

- Recommendation Phase: After the completion of the model training, the NCF model can be used to generate recommendations for users. In this phase, the algorithm uses the learned representations of users and products to predict the potential attractiveness of various products to specific users. Based on these estimations, a list of recommendations can be generated for users, suggesting products that will most likely interest them.

3. Conclusion

a. To sum up the knowledge gained from reading the article some advantages and disadvantages will be shown below.
   Let's start with the advantages:

   1. NCF uses matrix factorization and multi-layer perceptrons and considers linear and non-linear patterns when it comes to user-item actions. That may positively impact the recommendation accuracy in comparison with a more traditional approach.
   2. Adjustability and customization, for example with parameters, number of hidden layers, and choice of activation functions to come up with the best possible solution when it comes to recommendation.

   On the other hand, there are also some disadvantages to be aware of:

   1. Use of the neural networks might be challenging to work with. Implementation is complex and might take some time, or even the knowledge to implement such a complex solution.

   2. When it comes to usage of neural networks it works best when lots of data are available. Working with a smaller dataset might cause the ineffectiveness of this method

   3. There is also a concern when it comes to hyperparameter optimization. When talking about neural networks there is always a minor inconvenience which is the optimization of the hyperparameters. Done incorrectly it may hinder the model's efficiency.

b. A major difference that differentiates this solution from the previous ones is the usage of neural networks. As mentioned before it is a key factor for the analyzed papers creation. Some key aspects that are being improved with this method are mostly mentioned in the advantages above, but to be more precise creating the fundaments of neural network usage in recommendation systems which may lead to getting some attention and development of certain areas of interest.

c. This method is certainly an improvement in the recommendation systems, however, there are also a few downsides or limitations that may be worth mentioning. A few that were highlighted are:

   1. Complexity of the neural network approach. It is always hard to maintain, train, and optimize the model, not even to mention the amount of time that it takes to implement such. It also highlights the problem of computational power. It is also known that sometimes it might be overkill to create and train the model when the problem is not that complex and does not need such a solution. Sometimes understanding the recommendations that the model gives might be hard to understand in some scenarios and that is also worth mentioning.

   2. Huge amounts of data are needed in order to train and develop a model in such a way that it performs on a certain, expected level. It should be always taken into consideration if we have enough data to develop such a solution. Cold-start problems might also be quite limiting. What is the cold-start problem? It is the inability to draw inferences for users or items about which it has not yet gathered enough pieces of information.

d. The main problem that was presented in this article was the usability of the neural network to improve the field of recommendation systems, more specifically collaborative filtering that focuses on implicit feedback.
   The problem was solved successfully, and implementation and results were presented to confirm and document the improvement of collaborative filtering, however, the authors in the conclusion admit that there is room for improvement and that this article “opened up a new avenue of research possibilities for recommendation based on deep learning.”

e. Lasso regularization can be applied to the squared regression loss function to minimize the risk of obtaining a large variance of the result parameters. An example of the second term could be as on the formula below with λ parameter:

$$L_{sqr} = \sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} w_{ui}(y_{ui} - \hat{y}_{ui})^2,$$

$$L_{lasso}(\hat{\beta}) = \sum_{i-1}^{n}(y_i - x_i'\hat{\beta})^2 + \lambda\sum_{j-1}^{m}|\hat{\beta}_j|.$$