NCF-based Recommender System

Presenters: Abhi Mohnani, Chikai Shen, Earl Wu

Introduction

Recommender systems are everywhere.

- In recent years, deep learning has been widely applied in such systems
- Following a tutorial, we implemented a recommender system based on the Neural Collaborative Filtering (NCF) framework (https://www.kaggle.com/code/jamesloy/deep-learning-based-recommender-systems/notebook)
- We tested our implementation on an Amazon review dataset

What's NCF?

Neural Collaborative Filtering

- Proposed by He et al. in the paper with the same name from 2017
- Collaborative Filtering:
  - Making automatic predictions about user interests
  - By collecting preferences from many other users
- Before NCF:
  - Matrix Factorization (MF) is the state of the art
  - DNN was mostly used to model auxiliary information

[Submitted on 16 Aug 2017 (v1), last revised 26 Aug 2017 (this version, v2)]

Neural Collaborative Filtering

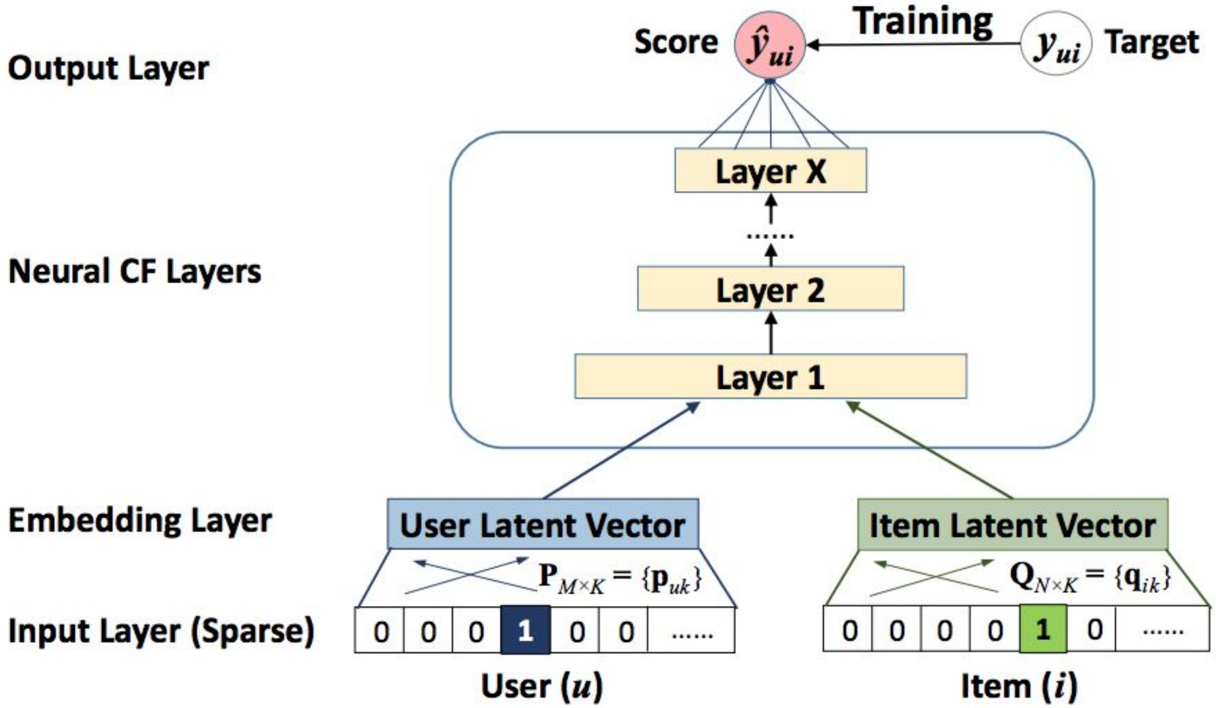
Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua

In recent years, deep neural networks have yielded immense success on speech recognition, computer vision and natural language processing. However, the exploration of deep neural networks on recommender systems has received relatively less scrutiny. In this work, we strive to develop techniques based on neural networks to tackle the key problem in recommendation -- collaborative filtering -- on the basis of implicit feedback. Although some recent work has employed deep learning for recommendation, they primarily used it to model auxiliary information, such as textual descriptions of items and acoustic features of musics. When it comes to model the key factor in collaborative filtering -- the interaction between user and item features, they still resorted to matrix factorization and applied an inner product on the latent features of users and items. By replacing the inner product with a neural architecture that can learn an arbitrary function from data, we present a general framework named NCF, short for Neural network-based Collaborative Filtering. NCF is generic and can express and generalize matrix factorization under its framework. To supercharge NCF modelling with non-linearities, we propose to leverage a multi-layer perceptron to learn the user-item interaction function. Extensive experiments on two real-world datasets show significant improvements of our proposed NCF framework over the state-of-the-art methods. Empirical evidence shows that using deeper layers of neural networks offers better recommendation performance.

Comments:	10 pages, 7 figures
Subjects:	**Information Retrieval (cs.IR)**
Cite as:	arXiv:1708.05031 **[cs.IR]**
	(or arXiv:1708.05031v2 **[cs.IR]** for this version)
	https://doi.org/10.48550/arXiv.1708.05031

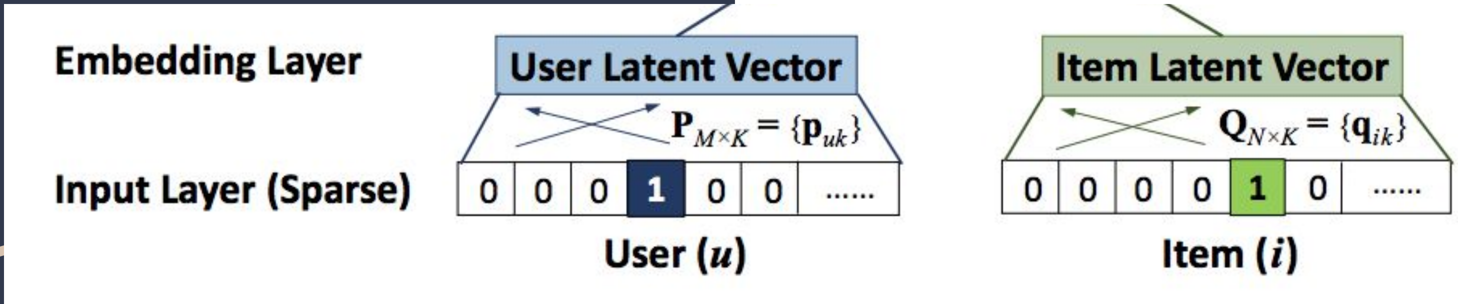
Reference paper by He et al.

Neural Collaborative Filtering Architecture



Embedding Layer

- The embedding layer allows us to “learn” the preferences of users and the properties of items.
- Each item and each user is mapped to a k-dimensional vector (k is a parameter of the model).
- These latent vectors represent what the model has learned about the item.



Our Dataset

Amazon product data

- Compiled by Julian McAuley @ UCSD
- Relevant information:
  - User id, Rating, ASIN
- # of entries: 22.5 million in total
- Data processing required:
  - User id and ASIN both contain special characters

```
ratings['userid'] = pd.factorize(ratings['userid'])[0]
ratings['ASIN'] = pd.factorize(ratings['ASIN'])[0]
```

  - Need to convert explicit data to implicit data

```
train_ratings.loc[:, 'rating'] = 1
```

userid	ASIN	rating	timestamp
A9DMTMLFR9CO5	1393774	5	1377907200
AHG1GTQZUYNJN	1393774	5	1372723200
A2TFO7NREP2B2D	1393774	5	1396396800
A2YAPAG1IPNK7K	1393774	5	1392422400
AEKGGV851HY3K	1393774	5	1130803200
A2MRQG8RN5JI7R	1393774	5	1396828800
A12R54MKO17TW0	1393774	5	1326067200
A1C7NPVPFF4OO8	1393774	5	1374364800
A22X72C51HQ7AS	1393774	5	1390262400

userid	ASIN	rating
6371	18446	1
7240	10118	1
35308	698	1
78734	28902	1
28609	3224	1

Example data

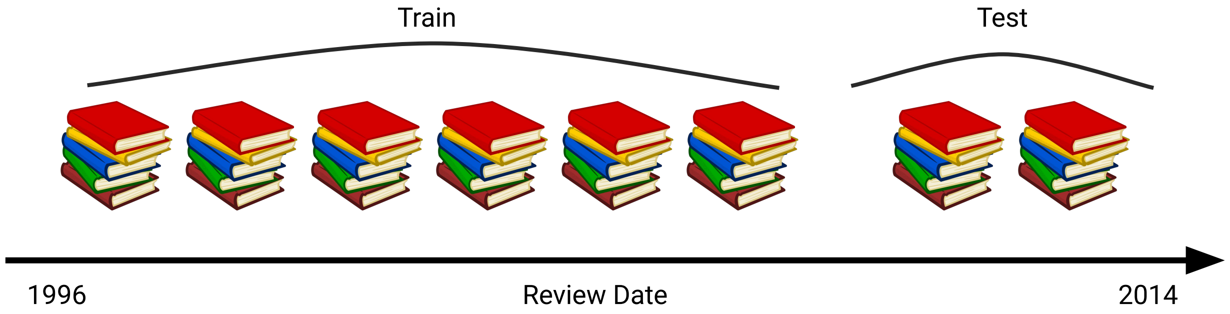
Train Test Split

Leave-one-out Split

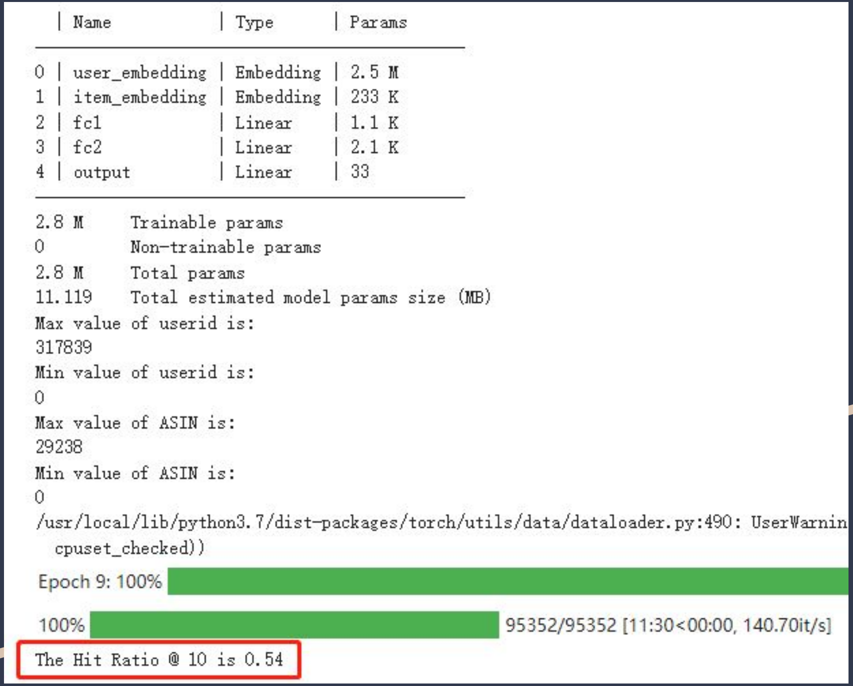
- The most recent review is used as the test set (i.e. leave one out), while the rest will be used as training data.

```
ratings['rank_latest'] = ratings.groupby(['userid'])['timestamp'].rank(method='first', ascending=False)

train_ratings = ratings[ratings['rank_latest'] != 1]
test_ratings = ratings[ratings['rank_latest'] == 1]
```



Results



```
  | Name           | Type      | Params
--------------------------------------------
0 | user_embedding | Embedding | 2.5 M
1 | item_embedding | Embedding | 233 K
2 | fc1            | Linear    | 1.1 K
3 | fc2            | Linear    | 2.1 K
4 | output         | Linear    | 33
--------------------------------------------
2.8 M     Trainable params
0         Non-trainable params
2.8 M     Total params
11.119    Total estimated model params size (MB)
Max value of userid is:
317839
Min value of userid is:
0
Max value of ASIN is:
29238
Min value of ASIN is:
0
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:490: UserWarnin
  cpuset_checked))
Epoch 9: 100%

100%  95352/95352 [11:30<00:00, 140.70it/s]
The Hit Ratio @ 10 is 0.54
```

Training information:

- **Number of epochs:** 10
- **Hit ratio:** 0.54
- **Potential improvement:**
    - Increase number of layers
    - Use dataset with actual implicit feedback
    - Change the ratio of negative to positive samples

Thank you!