Master's Project

Polina Berenstein
BGSE

Lívia Ónozó
BGSE

Jakob Pörschman
BGSE

1 Overview

2 Dataset

3 Identification strategy

3.1 Baseline

3.2 Neural Collaborative Filtering

The recommendation should take into account similar shoes, the customer has purchased in the past, and shoes similar users bought.

We have N users and M items in our database, from which we can define the user-item matrix as follows:

$$y_{ui} = \begin{cases} 1 & \text{if user } u \text{ bought item } i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $u = 1, ..., N$ and $i = 1, ..., M$

Since we do not have exact ratings from customers, we are using an implicit feedback matrix, that is if a customer buys a shoe, the system infers[1] that the customer is interested, which will result in a binary matrix.

Another issue arising from implicit matrix is that we do not observe negative feedbacks.

Similarities between users could be easiliy calculated (eg. cosine similarity, Jaccuard, etc.), but that would not guarantee the ordering, causing loss of ranking power. Thus we will use deep neural network to learn the collaborative filtering.

The input of the neural network is the binary representation of the user vectors and item vectors. Then we project this sparse representation into a dense matrix, as we can see below:

[1] In this case $y_{i,j} = 1$, however, it does not mean that the customer actually liked the shoes. One drawback of implicit feedback matrix is that it might generate noisy indication about preferences. Therefore one way one could consider is to use explicit matrix.

Only positive interactions:

	user_id	item_id	interaction
30718752	0.0	215.0	1.0
30718753	0.0	429.0	1.0
30718754	0.0	439.0	1.0
30718755	0.0	491.0	1.0
30718756	0.0	566.0	1.0
30718757	0.0	803.0	1.0
30718758	0.0	1420.0	1.0
30718759	0.0	2034.0	1.0
30718760	0.0	2110.0	1.0
30718761	0.0	2232.0	1.0
30718762	0.0	3495.0	1.0
30718763	0.0	3614.0	1.0
30718764	0.0	3669.0	1.0
30718765	0.0	3697.0	1.0
30718766	0.0	5353.0	1.0
30718767	1.0	856.0	1.0
30718768	1.0	1332.0	1.0
30718769	1.0	2093.0	1.0
30718770	1.0	3528.0	1.0
30718771	2.0	644.0	1.0

In the input layer both the user and item matrix are one-hot encoded, and then projected into the latent space. By multiplying the embedding layers we can use a multilayer network, in order to predict scores $(\hat{y}_u, i)$. The main goal here is to minimize the difference between $\hat{y}_u, i$ and y_u, i. Since we have a binary setting, we consider to using binary-crossentropy as in case of probabilistic problems. (and an Adam optimizer).

3.2.1 Matrix Factorization

In order to get the latent feature vectors we use matrix factorization, such that the inner product of the (normalized, unit length) latent vector for users (v_u) and for items (v_i) gives us the estimate of the user-item matrix.

$$\hat{y}_u, i = v_u^T v_i \tag{2}$$

In other words, we want to transform our matrix into a latent space, where the vectors represent user-item interactions. Linearly combining these vectors - assuming that dimensions of the latent space are independent of each other - will result in the estimate of the user-item

matrix. (MF is the linear model of the latent vectors.)

If the latent vectors are normalized, similarity between two users can also be measured with the cosine of angle of their latent vectors[2].

Matrix factorization basically consists of multiplying two submatrices to recunsruct the user-item matrix. The larger the value we get for the unobserved entries, the more likely that the corresponding user would be interested in buying the actual shoes.

References

[1] https://developers.google.com/machine-learning/recommendation/collaborative/basics

[2]Or equivalently, their inner product.