Recommender Systems HW2

Yishay Shapira 203016217
Or Sharir 201314796
Amir Wolfensohn 300339785

1) Write the objective of the model and explain its reasoning

We used the Bayesian Personalized Ranking (BPR) model.

While other recommender models seek to predict what rating each of the M users will give to each of the N items in order to minimize the error between the prediction rating and actual rating, the BPR model will predict the affinity between pairs of different items for each of the users- to put the items on the scale for the user preferences.
This is done by predicting the affinity relationships between given pairs in the data for each of the users.
We'll denote the affinity between a pair of items with $i >_u j$, meaning that for user u∈U item i is prefered over item j.

In this data we don't get an explicit feedback of ratings from the users and we need to observe their interactions with the items. We are getting signals, binary feedback whether the user has interacted with the specific item or not given options of items to interact with and he has only interacted with a single item in each pair.
by scaling the items by the predicted user preference one would be able to predict, given an instance of item appearances, which will be the item that the user will "pick" (buy, watch etc.)

the BPR model works as follows:

We denote Θ to be the parameters of the model that determine the personalized ranking.
Our model target is to find the probability that for user u, he will prefer item i over item j. fromally:

$$P(\boldsymbol{\Theta}|i >_u j) \alpha P(i >_u j|\boldsymbol{\Theta})P(\boldsymbol{\Theta})$$

This is a direct derivation of the Bayes theorem.
for calculating the likelihood , $P(i >_u j|)$, we use the sigmoid function:

$$P(i >_u j|\boldsymbol{\Theta}) = \boldsymbol{\sigma}(i >_u j|\boldsymbol{\Theta}) = \boldsymbol{\sigma}(\hat{r}_{u,i,j}|\boldsymbol{\Theta}) = \boldsymbol{\sigma}(\hat{r}_{u,i,j}(\boldsymbol{\Theta})) = \frac{1}{1+e^{-(\hat{r}_{u,i,j}(\boldsymbol{\Theta}))}} = \frac{1}{1+e^{-(\hat{r}_{u,i}(\boldsymbol{\Theta})-\hat{r}_{u,j}(\boldsymbol{\Theta})}}$$

when r^uij=r^ui−r^uj

for calculating the prior probability, p(Θ), we used p(Θ)=NVM(0,∑(Θ)) ⇒ Multivariate Normal distribution of Θ.

As seen in classroom, it can be written as:

$$\lambda_\Theta \|\Theta\|^2$$

now we will want to find the maximum for the posterior probability ,p(Θ|i>uj), for all tuples (u,i,j)in the data. so, assuming all probabilities are i.i.d (NVM), we can look at:

$$\prod_{u,i,j} p(i >_u j|\Theta)p(\Theta)$$

To simplify our calculations we will use the logarithmic function (which is valid as it is a monotonic transformation) :

$$ln\Big(\prod_{u,i,j} p(i >_u j|\Theta)p(\Theta)\Big)$$

which will get rid of the multiplications, and we'll get:

$$\sum_{u,i,j} ln\sigma\big(\hat{r}_{ui} - \hat{r}_{uj}\big) + lnp(\Theta)$$

and finally, to maximize it we'll have:

$$BPR = argmax_\Theta \sum_{(u,i,j)\in D} ln\sigma(\hat{r}_{ui}(\Theta) - \hat{r}_{uj}(\Theta)) - \lambda\|\Theta\|$$

2) Are you using negative examples? If so, what scheme have been employed to sample the negatives?

We don' t get explicit data, and we only get "positive" examples, meaning that from a pair of items we can only have the positive signals (Which of the items has the user chosen between given two items).

We used negative examples but they were created by us, based on assumptions made on this data- that items labeled with a "1" feedback, for any pairwise interaction, are to have a higher confidence that these were indeed "positive" feedbacks (User has prefered them in a specific pair). Using this level of confidence we then assumed that negative feedback will be for items that got labelled as "0" or missing items (Items not in any pair) for each user. It is possible that some of them will turn out to be picked by the customer eventually, but given that the data isn't explicit it is an assumption we had to take.

The scheme employed goes as follows:

1. to go over all the pairs in the data and for each user keeping a set of the "positive" items.
2. create a set containing all of the items per user
3. then we will create a list of "negative" items per user by calculating the outcome of: "all items per user" set - "user positive items"

3) Write the update step for each parameter

We used Stochastic Gradient Descent for the updating steps.
by using SGD we update each of the parameters like so:

$$x_u <- x_u + (\frac{1}{1+e^{(x_u v_i{}^T - x_u v_j{}^T)}} * (v_i - v_j) + x_u)$$
$$v_i <- v_i + (\frac{1}{1+e^{(x_u v_i{}^T - x_u v_j{}^T)}} * (x_u) + v_i)$$
$$v_j <- v_j + (\frac{1}{1+e^{(x_u v_i{}^T - x_u v_j{}^T)}} * (-x_u) + v_j)$$

This update is done for each of the tuples <u,i,j> at a time
* We eventually chose not to use the regularization in the code as we received better results without them

4) Write a pseudo code for the algorithm:

Bayesian Personalized Ranking
==========

1. Initialize biases λ , and set value to parameters $, K, N$
2. Initialize X and V, latent vectors
3. Store list of all users U in new Dataset D
4. For each user u∈U in D:
    a. Generate list P_u of **positive items** and store in **D**
    b. Generate list N_u of **Negative items** and store in **D**
5. Do For N iterations:
    a. For each combination of <u∈U, positive item∈LP_u> in D:
        i. sample Negative item from N_u
        ii. create sample_tuple <u∈U, positive item∈P_u, Negative item∈N_u> and store in S.
    b. Do For each sample_tuple <u,i,j> in S:
        i. SGD step- update for each of the parameter:
            1. $x_u <- x_u + (\frac{1}{1+e^{(x_u v_i{}^T - x_u v_j{}^T)}} * (v_i - v_j) + x_u)$
            2. $v_i <- v_i + (\frac{1}{1+e^{(x_u v_i{}^T - x_u v_j{}^T)}} * (x_u) + v_i)$
            3. $v_j <- v_j + (\frac{1}{1+e^{(x_u v_i{}^T - x_u v_j{}^T)}} * (-x_u) + v_j)$

c. $= * 0.9$
d. For each user u∈U do:
    i. For each item i of user u do:
        1. Calculate: rating= $x_u v_i^T$
        2. Add rating to u_ratings list
    ii. Add u to Users_ratings list
e. Create the prediction matrix R from Users_ratings list
f. Using R Calculate current iteration MPR vs Validation data
g. if MPR vs validation results were higher in the last run vs 2 previous runs:
    i. Break for loop //for early stopping
    ii.

* We eventually chose not to use the regularization in the code as we received better results without them

5) What hyper-parameters do you need to tune and how?

The hyper-parameters needed to be tuned are:

1. K- Number of dimensions for each of the latent vectors
2. −The Learning rate
3. $_{u,v,Bu}$ - L2 regularization parameters

all of them are to be tuned using a random search to find the best-performing model's parameters.
We run M different iterations for the hyper-parameter search.
For K we will sample randomly from the range of integers [5,40]
For −we will sample randomly from the range of integers [0.001,0.03]

Each iteration will run on our model with different values for K, and different initialization for $_{u,v,Bu}$.
Then, we will keep and compare all of the resulting models MPR and Val_MPR, and also the Precision@K parameters.
After finishing on running all of the models for the random search, we then are able to retrieve the best performing models hyper-parameters settings for our usage.

* We eventually chose not to use the regularization in the code as we received better results without them

6) Do you need a validation set? If so, how would you create it? How would you check for convergence?

We need a validation set, we split from the train data 20% for validation, to make sure we have at least one record for each user in the validation data we split the data in a stratified fashion and we used the user ID as the class label.

```
12 #split to train and val and make sure all the users in the val df too (using stratify=data['UserID'])
13 train_data, val_data = train_test_split(data, test_size=0.20, random_state=27, stratify=data['UserID'])
```

We used the validation set for two purposes:

- We implemented random search in order to find the best hyper-parameters for our model, we used the MPR score of the validation set to rank the model of each evaluation in the random search.
- We checked for convergence using the validation set, we performed early stopping, in case the MPR score of the validation set increased twice in a row, we stopped the evaluation and reverted the updates of U and V vectors from the last two updates.

7) How would you train the last \ best model?

After we performed the random search and found the best hyper-parameters, we trained again the U and V vectors on the original train data we have (ob both train and validation sets).

8) Implement your model. Explain the main work items you had to take

We took the train set and split it into 80% train and 20% validation.
The train set is composed of pairs of user and an item he saw and in each BPR running we are sampling again to each pair of the train set another item that the user didn't see.
We are running the BPR for 20 epochs and on each epoch we are running the BPR on the train set (as described above, in each epoch the train set the BPR is running on will be different because in each epoch we are sampling again an item the user didn't saw) and checking MPR, K Precision measures on the validation set with an early stopping to the epochs that if we have a consecutive 2 epochs with increasing MPR value then we are stopping the epochs then and taking the epoch before that 2 epochs and considering him to be our 'best model'.

All what was described so far is being tested for 3 evaluations that on each evaluation we are randomly sampling the hyper parameters (K-latent vector, learning rate) and take the best model out of the 3.

After we have the best model hyperparameters we are taking our entire train set (from before the split of train and validation) and running on it with the best model's

hyperparameters for 30 epochs or an early stopping as was described above and after we have a trained model we are running it on the test sets and outputting to excel.

9) What is the Precision@K)K=1, K=10, K=50) and the MPR of your model?

After the random search we got our best model and when he was tested on the validation set (after 20 epochs- we had no early stopping in our 'best model') we got-

- MPR - 0.137636
- Precision@K=1 - 0.1288
- Precision@K=10 - 0.1105
- Precision@K=50 - 0.089

Here is the output from the code-

```
The bast model is:
 {'K': 20, 'learning_rate': 0.022559252043041548, 'val_mpr': 0.13763604768204668, 'val_precision_k_1': 0.12880794701986756, 'va
l_precision_k_10': 0.11054635761589862, 'val_precision_k_50': 0.08919205298013334}
```

After we run again the model on the original train set (from before the split into train and validation) with our best model hyperparameters for 30 epochs or early stopping (run all the 30 epochs, no early stopping) we got -

- MPR - 0.12627
- Precision@K=1 - 0.6238
- Precision@K=10 - 0.54995
- Precision@K=50 - 0.449

Here is the output from the code-

```
Iteration: 30 ; mpr = 0.1262751575 ; precision_k_1 = 0.6238410596 ; precision_k_10 = 0.5499503311 ; precision_k_50 = 0.44912913
91
```

10) Submit the test result files according to the instructions above. Also note these additional instructions

11) You also need to submit your python code along with the report.