Information Retrieval and Recommender Systems

Homework 2 Instructions

1 General description

1.1 In this homework we will build and compare few recommender system algorithms

The first category is simple baselines, the second is neighborhood-based and the last is matrix factorization. For each approach, we will estimate the parameters of the model on the train set and compute the root mean squared error (RMSE) on the test set.

1.2 Notation standarts

- R: the set of all ratings
- U: the set of all users, u and v denotes users
- I: the set of all items. i and j denotes items
- r_{ui}: the true rating of user u for item i
- $\hat{r_{ui}}$: the estimated rating of user u to item i
- b_{ui}: the baseline rating of user u for item i
- μ: the mean of all ratings
- μ_u: the mean of all ratings
- μ_i: the mean of all ratings
- $N_u^k(i)$: the k nearest neighbors of item i that are rated by user u. This set is computed using a similarity metric

1.3 Baselines:

- `Simple mean`: - for each user (in the train set..) estimate one parameter, the mean of his rankings.
  - Prediction :
    $$\hat{r_{ui}} = \mu_u$$
  - Number of estimated parameters $|U|$
  - Estimate the parameters by computing the means from the data.
- `linear regression baseline`: - for each user and item estimate one parameter.
  - Prediction :
    $$\hat{r_{ui}} = b_{ui} = \mu + b_u + b_i$$
  - Number of estimated parameters $|U| + |I| + 1$
  - Estimate μ by computing the mean from the data and estimate b_u and b_i by SGD that minimizes the least squres formula with regularization -

$$min_{b_u \in U, b_i \in I}[\Sigma_{(u,i \in R_{train})}(r_{ui} - \mu + b_u + b_i)^2 + \gamma \cdot (\Sigma_u\ b_u^2 + \Sigma_i\ b_i^2)]$$

1.4 Neighbourhood models:

- `Simple KNN`: - with item based similarities.
  - s(i,j) - similarity between item i and item j
  - Number of estimated parameters $< 0.5 * I * (I - 1)$
  - Prediction :

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} \text{sim}\,(i,j) \cdot r_{uj}}{\sum_{j \in N_u^k(i)} \text{sim}\,(i,j)}$$

- `Baseline KNN`: - using item similarities and b_{ui} that were computed in the baseline step.
  - Prediction :

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in N_u^k(i)} \text{sim}\,(i,j) \cdot \left(r_{uj} - b_{uj}\right)}{\sum_{j \in N_u^k(i)} \text{sim}\,(i,j)}$$

Remarks - in this model we don't use s(i,j) <= 0. We will use pearson correlation as a similarity metric.

1.5 Matrix factorization:

- `matrix factorization`: baseline model with hidden vectors that describe latent features (dimension k. k is an hyper-parameter)
  - Prediction :

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

  - Number of estimated parameters $(|\,I\,| + |\,U\,|) * (k + 1) + 1$
  - Estimate μ by computing the mean from the data and estimate b_u, b_i p_u, q_i by SGD that minimizes the least squres formula -

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \gamma \cdot (\Sigma_u\ b_u^2 + \Sigma_i\ \ b_i^2 + \Sigma_u \|p_u\|^2 + \Sigma_i\ \|q_i\|^2)$$

It is recommended to read the original paper.

Note - we learn the parameters jointly and not use b_u, b_i from previous algorithms.

2 In depth description

Each model will be implemented in a separate file. The code and the generated files should be submitted and **run perfectly** on the checker computer. therefore, put the data in the data folder and **do not deviate** from the filenames in the `config.py` You can't use external libraries to fit the algorithm (you should implement the algorithms by yourselves)- but you can use numpy, scipy, pandas, and libraries like those.

1. Each model suppose to implement `fit()` and `calculate_rmse()` functions. Make sure that your code is readable and super clear. We attach an example script that you can use as a template and change whatever you want for this purpose in `linear_regression_baseline.py` also, we attach an interface for the algorithms.

2. Implement a function `train, validation = get_data()` that reads the data from the files and returns the train and validation files. The function should fix the user_id and item_id to be pythonic (reindex the user id's and item id's to start from 0 to n-1).

3. In `linear_regression_baseline.py` save the parameters of the model to BASELINE_PARAMS_FILE_PATH so we can upload it in `knn_baseline.py`. Print the value of the objective function and train RMSE each epoch. You don't need to implement a function that uploads those parameters if already exist.

4. In `knn.py` save the correlations to CORRELATION_PARAMS_FILE_PATH . Make sure that the file size is less the 50MB so you can compress it (zip) and attach to your solution. In order to do so we saved the items columns in int16 format and the similarity column in float32 format. Make sure that in you implement save_params and upload_params so we can run it when we examine your code. In simple words, make sure that in the `fit()` function you have something like this:

```
if Path(CORRELATION_PARAMS_FILE_PATH).exists():
    self.upload_params()
else:
    self.build_item_to_itm_corr(train) # the name we gave for
the function
    self.save_params()
```

Make sure to estimate the parameters in a way that scales well (iteratively, do not assume that everything will fit in memory like df.corr() (df: pandas DataFrame)). **Use tqdm so we can see how much time it took you to run it** Estimating the correlations suppose to take around half an hour. Make sure to save only positive correlations.

5. In `knn_baseline.py` make sure that you upload the parameters from `linear_regression_baseline.py` and `3.knn.py`.

6. in `matrix_factorization.py` we don't upload any weights and do not save any weights. Make sure you understand well the update steps and how it was derived in order implement it well. Print the objective function value and the train RMSE at each epoch. Add to the pdf the deriviatives per each parameter.

7. It may be that in the validation dataset there will be items/users that we have not seen in the training set. you should deal with it wisely.

2.1 General instructions

1. Students will form teams of three people each and submit a single homework for each team in the format - ID1_ID2_ID3.ipynb
2. Groups of four are not allowed and you are more than welcome to form groups of two.
3. **Do not write your names anywhere.**
4. The same score for the homework will be given to each member of the team.

5. The goal of this homework is to test your understanding of the concepts presented in the lectures.
   If a topic was not covered in detail during the lecture, you are asked to study it online on your own. Anyhow, we provide here detailed explanations for the code part and if you have problems - ask.
6. Questions can be sent to the forum, you are encouraged to ask questions but do so after you have been thinking about your question. Since it is the first time we provide this homework please notify us if there is a bug/something is unclear, typo's exc..

7. We use Python 3.7 for programming.

2.2 Submission guidelines

For each algorithm, write how you implemented it efficently and in which data structures you used for this purpose ,add a table with the RMSE of the validation set RMSE for each algorithm (pdf file, up to 2 pages). Submit the code, the generated files and the pdf in **ID1_ID2_ID3.zip**