Assignment on Matrix Factorization

CSE4238: Soft Computing Lab

Instructor: Nibir Mandal

1 Matrix Factorization

In this assignment, you will build a matrix-factorization model using standard python library. Then you will apply this model to a dataset and evaluate the performance of the model. Finally, you will analyze the results of the model.

2 Dataset

2.1 Dataset Creation

- Download datasets from the following links- index_id.csv, Matrix_Factorization_Assignment.csv
- Find your index id from index_id.csv file
- Remove all rows that satisfy row_index % your_index_id = 0 and all columns that satisfy column_index % your_index_id = 0 *[Use pandas DataFrame to accomplish this step. Do not change row or column index.]*

2.2 Validation Dataset

This section is optional. You can split the dataset and hold a few data for validation purposes.

3 Experiments

3.1 Implementation

Let assume $X = UV^T$ where dimension of $X,\ U,\ V$ are $N * M,\ N * K,\ M * K$ respectively. Implement the following algorithm using python library-

1. Initialize $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3,\ \ldots, \mathbf{u}_N$ matrix randomly
2. Normalize U matrix- i.e., $||U|| = 1$
3. Initialize $\lambda_u\ and\ \lambda_v$ with $(0.00015 + 0.0001 * (group\_id\ \%\ 8))$ and $(0.00025 - 0.0001 * (group\_id\ \%\ 7))$ respectively
4. Update each column latent factor $v_1, v_2, v_3,\ \ldots, v_M$ as follows

$$\mathbf{v}_m^* = (\sum_{n \in \Omega_{c_m}} \mathbf{u}_n \mathbf{u}_n^T + \lambda_v I_K)^{-1} \sum_{n \in \Omega_{c_m}} x_{n,m} \mathbf{u}_n$$

5. Update each column latent factor $u_1, u_2, u_3,\ \ldots, u_N$ as follows

$$\mathbf{u}_n^* = (\sum_{m \in \Omega_{r_n}} \mathbf{v}_m \mathbf{v}_m^T + \lambda_u I_K)^{-1} \sum_{m \in \Omega_{r_n}} x_{n,m} \mathbf{v}_m$$

6. Calculate mean squared error as follow

$$L = \frac{\sum_{(n,m) \in \Omega_{valid}} (x_{n,m} - \mathbf{u}_n \mathbf{v}_m^T)^2}{\#\ of\ valid\ values}$$

7. Compare loss, L, between consecutive iterations and stop the iteration if improvement of L is not notable otherwise repeat from step 4. *[Stop iteration after 800 or 1000 iteration if convergence is too slow]*

Here, even group students will follow the algorithm as stated above. For odd group students, there are slight changes. They will initialize v (instead of u) in step 1, normalize it in step 2, update u matrix in step 4, and update v in step 5.

3.2 Hyperparameter Tuning

In this assignment, you will tune two hyperparameters- K and #iterations. You can follow any strategy to tune these hyperparameters.

4 Results

4.1 Report Generation

Generate the following reports-

- Generate Loss curves for the hyperparameters- i.e., K and #iterations
- Report the optimal hyperparameters with proper explanation
- Calculate cosine similarity between all pairs of users (i.e., $u_1,\ u_2,\ \dots, u_N$)
- Calculate cosine similarity between all pairs of movies (i.e., $v_1,\ v_2,\ \dots, v_N$)
- Suggest five movies(v) that user u did not review yet
- Discuss the results of your model with suitable diagram or graph or table
- Write down a few applications of this study

5 Assignment Submission

Follow these guidelines-

- Create a Kaggle/Colab/jupyter notebook and complete the coding part of this assignment
- Download the notebook as .pynb or .py extensions and rename the file as 'yourId_yourIndexId_codes.(pynb/py)'
- Prepare all reports and then create a pdf file containing all of them
- Rename the pdf file as follows: 'yourId_yourIndexId_reports.pdf'
- Submit these two files in google classroom
- Submission Dateline: **11:59 PM,** 17^{th} **July 2021**
- DO NOT COPY FROM YOUR FRIEND. Plagiarism is strictly prohibited and punishable.