Your Name: seowoo Han
Your Andrew ID: seowooh
Your Nickname on Leaderboard: khan5555

Homework 3

0. Statement of Assurance

1. Did you receive any help whatsoever from anyone in solving this assignment? Yes / No.
If you answered 'yes', give full details? (e.g."Jane explained to me what is asked in Question 3.4").

Insoo Kim pointed and explained to me about Question 2.

Byeongju Han pointed to me about Question 2.4.

Sooah Lee, Minkyung Kim gave me an insight into this assignment.

2. Did you give any help whatsoever to anyone in solving this assignment? Yes / No.
If you answered 'yes', give full details? (e.g. "I pointed Joe to section 2.3 to help him with Question 2").

3. Did you find or come across code that implements any part of this assignment? Yes / No.
If you answered 'yes', give full details? (e.g. book & page, URL & location within the page, etc)

https://medium.com/@rabinpoudyal1995/nearest-neighbour-based-method-for-collaborative-filtering-16961c962dd

https://www.dataquest.io/blog/k-nearest-neighbors-in-python/

https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-1-knn-item-based-collaborative-filtering-637969614ea

https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/

http://na-o-ys.github.io/others/2015-11-07-sparse-vector-similarities.html

1. Corpus Exploration (10)

Please perform your exploration of the training set.

1.1 Basic statistics (5)

Statistics	
the total number of movies	5353
the total number of users	10858
the number of times any movie was rated '1'	53852
the number of times any movie was rated '3'	260055
the number of times any movie was rated '5'	139429
the average movie rating across all users and movies	3.381

For user ID **4321**	
the number of movies rated	73
the number of times the user gave a '1' rating	4
the number of times the user gave a '3' rating	28
the number of times the user gave a '5' rating	8
the average movie rating for this user	3.151

For movie ID **3**	
the number of users rating this movie	84
the number of times the user gave a '1' rating	10
the number of times the user gave a '3' rating	29
the number of times the user gave a '5' rating	1
the average rating for this movie	2.524

1.2 Nearest Neighbors (5)

	Nearest Neighbors
Top 5 NNs of user 4321 in terms of dot product similarity	1: 980 2: 551 3: 2586 4: 3760 5: 90
Top 5 NNs of user 4321 in terms of cosine similarity	1: 8497 2: 9873

	3: 7700 4: 8202 5: 3635
Top 5 NNs of movie 3 in terms of dot product similarity	1: 1466 2: 3688 3: 3835 4: 4927 5: 2292
Top 5 NNs of movie 3 in terms of cosine similarity	1: 5370 2: 4857 3: 5391 4: 4324 5: 5065

2. Rating Algorithms (50)

2.1 user-user similarity (10)

Rating Method	Similarity Metric	K	RMSE	Runtime(sec)
Mean	Dot product	10	1.0024	51.268
Mean	Dot product	100	1.0067	82.125
Mean	Dot product	500	1.0430	217.298
Mean	Cosine	10	1.0024	93.399
Mean	Cosine	100	1.0067	124.183
Mean	Cosine	500	1.0430	259.405
Weighted	Cosine	10	1.1463	94.233
Weighted	Cosine	100	1.1766	129.028
Weighted	Cosine	500	1.1788	281.305

2.2 Movie-movie similarity (10)

Rating Method	Similarity Metric	K	RMSE	Runtime(sec)
Mean	Dot product	10	1.0207	16.899
Mean	Dot product	100	1.0467	20.807
Mean	Dot product	500	1.1109	65.942
Mean	Cosine	10	1.0207	32.191
Mean	Cosine	100	1.0467	36.265
Mean	Cosine	500	1.1109	81.495
Weighted	Cosine	10	1.1478	32.493
Weighted	Cosine	100	1.1772	38.457
Weighted	Cosine	500	1.1790	118.864

2.3 Movie-rating/user-rating normalization (10)

Rating Method	Similarity Metric	K	RMSE	Runtime(sec)

Mean	Dot product	10	0.992	125.106
Mean	Dot product	100	0.996	165.799
Mean	Dot product	500	1.012	364.607
Mean	Cosine	10	0.992	230.941
Mean	Cosine	100	0.996	275.580
Mean	Cosine	500	1.012	473.011
Weighted	Cosine	10	0.991	231.355
Weighted	Cosine	100	0.993	275.279
Weighted	Cosine	500	1.006	631.069

Add a detailed description of your normalization algorithm.

Reconstructed the train set is the csr_matrix format. csr_matrix is row users and col movies. After the sum of each row, the value was divided by the total number of movies to find the average for each user. Then, the average for each user was divided by the rating given by that user. The average and norm of the ratings given by the input user are stored in memory. Multiply the predicted rating by the norm of the input user, and add the average. I weighted this value and the average rating of that movie to 0.6, 0.4, respectively.

2.4 Matrix Factorization (20)

a. Briefly outline your optimization algorithm for PMF
   I used the formula in the paper as it is. Normalization is not used to use this method. Because I set the score between 0 and 1 in the paper, I set all the scores between 0 and 1 by subtracting 1 and dividing by 4 from the original train score. I used GPU(RTX 2080Ti) to reduce computation time and PyTorch as a library. And I used Adam optimizer to minimize RMSE. I tested with varying learning rates and iterations. The combination of learning rate and iteration has limited performance. So I've weighted three loss functions. The weight was also tested in various combinations, and the weight was used as the value showing the best performance.

b. Describe your stopping criterion
   It did not set the exact stopping criterion. I tried several times and adjusted the stopping criterion for the best results. We experimented with not only adjusting the iteration but also the learning rate.

Num of Latent Factors	**RMSE**	**Runtime(sec)***	**Num of Iterations**
2	0.954	27.986	2000
5	0.923	25.181	2000
10	0.925	25.383	2000
20	0.936	26.044	2000
50	0.955	28.148	2000

3. Analysis of results (15)

Discuss the complete set of experimental results, comparing the algorithms to each other. Discuss your observations about the various algorithms, i.e., differences in how they performed, what worked well and didn't, patterns/trends you observed across the set of experiments, etc. Try to explain why specific algorithms or approaches behaved the way they did.

Matrix factorization showed the best performance. Because unlike other methods, this method can be optimized through several iterations to improve performance. That is, the hyperparameter can be adjusted in the direction of reducing the error. I've tested the learning rate and iteration over and over again and set the parameters that gave the best performance. The disadvantage is that the runtime takes a long time. The neighborhood method is the concept of solving a recommendation problem by finding a local structure. Conversely, Matrix Factorization is a concept to solve the recommendation problem by finding a global structure. The concept of finding a global structure is more accurate than a local solution.

The next best performance is the Pearson correlation coefficient (PCC). Vector normalization can be used to compare the bias between user and movie. When using the PCC method, I reconstructed the bias of one user for the final prediction. Bias is a personal feature and should not be ignored. Due to the constraints of this task, experiments 1 and 2 were difficult to manipulate to improve performance. I found both user-user similarity and movie-movie similarity in Experiment 3. User-user similarity results in less RMSE. Thus I got results for various experiments with user-user similarity. Considering only the user bias, the results were worse than experiments 1,2. Therefore, we consider the average of user-bias and specific movie. The performance was improved by weighting the value considering user bias and the average of a specific movie by 0.6, 0.4 separately.

Experiments 1 and 2 did not make a big difference. The difference between the two is whether you use dot or cosine to find similarity. If you divide the norm of a vector to find cosine similarity, the dot, and the equation are the same. To make a slight difference, cosine similarity was obtained with the norm of the absolute value of the vector. Since Experiments 1 and 2 did not use normalization, they did not consider the bias of the user and movie. Therefore, the experiments have the lowest performance among the four experiments.

Finally, user-user similarity using the Matrix Factorization algorithm resulted in lower RMSE. The number of latent factors is 2, iteration is 2000, and the learning rate is 0.003.

4. The software implementation (5)

Add detailed descriptions of software implementation & data preprocessing, including:

1. A description of what you did to preprocess the dataset to make your implementations easier or more efficient.

If no user or movie exists in the train set, I used shape, a parameter of csr_matrix. I created csr_matrix using the user max and movie max values from the Train set. csr_matrix assigns a value of 0 to that column or row if there is no user or movie.

2. A description of significant data structures (if any); any programming tools or libraries that you used;

Used libraries: scipy, numpy, sklearn, time, pandas

I used Pytorch for question 2.4.

Detailed libraries are listed in requirements.txt.

The program consists of:

main.py: Save train, dev, and test set and run the test, q1, and q2 functions.

test.py: Contains functions for creating "test-prediction.txt" and "dev-prediction.txt" files.

q1.py: This file contains the functions for q1.

q2.py: This file contains the functions for q2.

q2_4.py: This file contains the function for question 2.4.

KNN_utils.py: This file contains a function to obtain KNN and a function to solve q1 and q2.

KNN algorithm is implemented by myself.

3. Strengths and weaknesses of your design, and any problems that your system encountered;

Strengths: You can select user-based / item-based, dot / cosine, weight_mean / mean using function parameters in a single function. So the code is neat.

Weakness: I didn't make my code for saving as a text file neatly. My code is complex by implementing hard coding.