Uzi Smadja
usmadja

Homework 4

0. Statement of Assurance

You must certify that all of the material that you submit is original work that was done only by you. If your report does not have this statement, it will not be graded.

All work was done by me.

1. Corpus Exploration (8%)

Please perform your exploration on the training set.

1.1 Basic statistics (4%)

Statistics	
the total number of movies	5392
the total number of users	10916
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	3.38

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	3.15

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	2.5

1.2 Nearest Neighbors (4%)

	Nearest Neighbors
Top 5 NNs of user 4321 in terms of dot product similarity	262 169 411 155 980
Top 5 NNs of user 4321 in terms of cosine similarity	7474 8527 9303 7415 8249
Top 5 NNs of movie 3 in terms of dot product similarity	1904 3386 5216 4491 1873
Top 5 NNs of movie 3 in terms of cosine similarity	452 4680 3804 3877

2. Basic Rating Algorithms (40%)

2.1 User-user similarity

Rating Method	**Similarity Metric**	**K**	**RMSE**	**Runtime(sec)***
Mean	Dot product	10	1.0023	55
Mean	Dot product	100	1.0067	81
Mean	Dot product	500	1.0429	163
Mean	Cosine	10	1.0631	52
Mean	Cosine	100	1.0619	63
Mean	Cosine	500	1.0753	107
Weighted	Cosine	10	1.0628	52
Weighted	Cosine	100	1.0614	61
Weighted	Cosine	500	1.0739	102

*runtime should be reported in seconds.

2.2 Movie-movie similarity

Rating Method	Similarity Metric	K	RMSE	Runtime(sec)
Mean	Dot product	10	1.0207	51
Mean	Dot product	100	1.0468	145
Mean	Dot product	500	1.1108	345
Mean	Cosine	10	1.0174	44
Mean	Cosine	100	1.0639	106
Mean	Cosine	500	1.1183	275
Weighted	Cosine	10	1.0174	51
Weighted	Cosine	100	1.0639	121
Weighted	Cosine	500	1.1183	330

2.3 Movie-rating/user-rating normalization

Rating Method	Similarity Metric	K	RMSE	Runtime(sec)
Mean	Dot product	10	0.9518	53
Mean	Dot product	100	0.9719	90
Mean	Dot product	500	1.0007	213
Mean	Cosine	10	0.9647	90
Mean	Cosine	100	0.9871	155
Mean	Cosine	500	1.0080	312
Weighted	Cosine	10	0.9621	66
Weighted	Cosine	100	0.9828	88
Weighted	Cosine	500	1.0023	321

Add a detailed description of your normalization algorithm.

I got the idea from this paper: [https://datajobs.com/data-science-repo/Collaborative-Filtering-[Koren-and-Bell].pdf](https://datajobs.com/data-science-repo/Collaborative-Filtering-[Koren-and-Bell].pdf)

Where I normalized the ratings in the following way:

$$\frac{r(u_i, M_j) - \mu_i}{\sigma_i}$$

Where μ_i is the mean of all of user i's ratings and σ_i is the standard deviation of the user's ratings. Next step is calculating the predicted score over the k nearest neighbors (weighted or mean), the new score would be $knn_{score}(u_i, M_j) \cdot \sigma_i + \mu_i$ in order to re-assess the score. I also added a boundary verification in case the new scores are higher than 5 or lower than 1

2.4 Bipartite clustering information

Running time of bipartite clustering in seconds: 50

Total number of user clusters: 1000

Total number of item clusters: 500

How did you pick the number of clusters?

I noticed I was getting better RMSE score for larger number of clusters but they took a lot more time to process. I would have chosen a larger number to get better RMSE but the computation time was too long.

2.5 User-user similarity

Rating Method	Similarity Metric	K	RMSE	Runtime(sec) *
Mean	Dot product	10	1.1212	213
Mean	Dot product	100	1.1259	261
Mean	Dot product	500	1.1314	402
Mean	Cosine	10	1.1248	219
Mean	Cosine	100	1.1290	286
Mean	Cosine	500	1.1319	428
Weighted	Cosine	10	1.1128	151
Weighted	Cosine	100	1.1261	178
Weighted	Cosine	500	1.1533	320

*runtime should be reported in seconds. Do not include the running time for the bipartite clustering in this column.

2.6 Movie-movie similarity

Rating Method	Similarity Metric	K	RMSE	Runtime(sec) *
Mean	Dot product	10	1.1507	581
Mean	Dot product	100	1.1567	643
Mean	Dot product	500	1.1675	779
Mean	Cosine	10	1.1526	250
Mean	Cosine	100	1.1556	278
Mean	Cosine	500	1.1682	311
Weighted	Cosine	10	1.1537	179
Weighted	Cosine	100	1.1579	229
Weighted	Cosine	500	1.1695	324

*runtime should be reported in seconds. Do not include the running time for the bipartite clustering in this column.

4. Analysis of results (20%)

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 certain algorithms or approaches behaved the way they did.

The Movie-biased PCC-based algorithm with dot product similarty and mean rating with k of k nearest neighbors set to 10 got the best RMSE score of 0.9518

While the worst (excluding the BPC) RMSE score of 1.1183 is for movie-movie similarity with dot product and mean with k set to 500 nearest neighbors.

It seems as if the larger the K, the larger(worse) the RMSE score. The reason of this is because 500 nearest neighbors on a set of around 10000 people does not give neighbors that are similar enough and we get neighbors that affect the prediction who are not similar at all.

Movie-movie similarity performs better with the PCC-based experiment than user-user because movies are more biased than users . Bad movies will always get bad scores wheras users who only give low ratings are rare/nonexistent.

The weighting scheme does not seem to affect the RMSE.

4. The software implementation (15%)

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

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

I did not preprocess the dataset.

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

I used Numpy and scipy.sparse - csr_matrix to represent sparse matrices.

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

At first I encountered a problem with my PCC-based experiment when I didn't check that the new scores at the end were between 1 and 5. Because of the division by standard deviation, the subtraction of the mean, and the inverse of that at the end, it is possible to get scores that are below 1 or higher than 5.

I noticed that not excluding the user from its own neighbors results in bad RMSE score.

The bipartite reinforcement algorithm took a lot longer than experiments 1-2. I assume it is a combination of large clusters making calculations longer and non ideal bipartite reinforcement optimizations (from hw2).