Liruoyang Yu

liruoyay

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.

I certify that all of the materials that I submit is my original work that was done by me and only 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	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	3.38

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	2.52

1.2 Nearest Neighbors (4%)

	Nearest Neighbors
Top 5 NNs of user 4321 in terms of dot product similarity	980, 551, 3760, 2586, 90
Top 5 NNs of user 4321 in terms of cosine similarity	8497, 9873, 7700, 8202, 3635
Top 5 NNs of movie 3 in terms of dot product similarity	1466, 3688, 3835, 4927, 2292
Top 5 NNs of movie 3 in terms of cosine similarity	5370, 4857, 5391, 4324, 5065

2. Basic Rating Algorithms (40%)

2.1 User-user similarity

Rating Method	**Similarity Metric**	**K**	**RMSE**	**Runtime(sec) ***
Mean	Dot product	10	1.002	13.5
Mean	Dot product	100	1.007	44.5
Mean	Dot product	500	1.043	178.8
Mean	Cosine	10	1.063	14.3
Mean	Cosine	100	1.062	45.2
Mean	Cosine	500	1.075	173.5
Weighted	Cosine	10	1.063	15.4
Weighted	Cosine	100	1.062	50.5

Weighted	Cosine	500	1.072	201.8

*runtime should be reported in seconds.

2.2 Movie-movie similarity

Rating Method	Similarity Metric	K	RMSE	Runtime(sec)
Mean	Dot product	10	1.021	4.9
Mean	Dot product	100	1.047	40.3
Mean	Dot product	500	1.111	199.7
Mean	Cosine	10	1.017	4.5
Mean	Cosine	100	1.064	35.6
Mean	Cosine	500	1.118	173.2
Weighted	Cosine	10	1.015	5.4
Weighted	Cosine	100	1.057	39.4
Weighted	Cosine	500	1.102	195.3

2.3 Movie-rating/user-rating normalization

Rating Method	Similarity Metric	K	RMSE	Runtime(sec)
Mean	Dot product	10	1.025	4.0
Mean	Dot product	100	1.070	35.6
Mean	Dot product	500	1.122	170.9
Mean	Cosine	10	1.025	4.0
Mean	Cosine	100	1.070	35.6
Mean	Cosine	500	1.122	170.9
Weighted	Cosine	10	1.022	4.6
Weighted	Cosine	100	1.063	41.0
Weighted	Cosine	500	1.106	198.9

Add a detailed description of your normalization algorithm.

I normalized each row of item so that the mean of each row is 0 and 2L-norm is 1.

Then I used model-based CF predicting future ratings.

The time I reported above doesn't include the time to compute the model offline, which took 347.038496017 sec.

2.4 Bipartite clustering information

Running time of bipartite clustering in seconds: total 600s for 5 times

Total number of user clusters: 1000

Total number of item clusters: 600

How did you pick the number of clusters?

Set it to be greater than 500, and experiment.

2.5 User-user similarity

Rating Method	Similarity Metric	K	RMSE	Runtime(sec) *
Mean	Dot product	10	1.170	11.5
Mean	Dot product	100	1.178	42.7
Mean	Dot product	500	1.179	175.1
Mean	Cosine	10	1.171	11.8
Mean	Cosine	100	1.178	42.3
Mean	Cosine	500	1.179	174.1
Weighted	Cosine	10	1.169	11.8
Weighted	Cosine	100	1.170	42.8
Weighted	Cosine	500	1.170	175.3

*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	1.177	8.9
Mean	Dot product	100	1.179	39.3
Mean	Dot product	500	1.179	169.5
Mean	Cosine	10	1.177	8.9
Mean	Cosine	100	1.179	39.3
Mean	Cosine	500	1.179	169.7
Weighted	Cosine	10	1.177	8.9
Weighted	Cosine	100	1.177	39.6
Weighted	Cosine	500	1.177	170.8

*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.

What worked:

1. Weighted mean. This worked because the predicted rating tends to be influenced more by the users/items that they're closer to.
2. Offline model computation. It helped reduce the response time for predicting, because apparently the computation is much less than memory-based method.

What didn't work:

1. Cosine similarity. The results were pretty much the same for dot product similarity and cosine similarity. I think it's because that cosine similarity just acted as a form of normalization compared to dot product similarity.
2. Increasing K. I think it didn't work because when the size of neighborhood grew, the results were influenced by too many users/items that were not similar to the test points.
3. Bipartite clustering. I think the reason that this didn't work is same as increasing K, since clustering indirectly increased the data points that affected the predictions.
4. PCC. It did affect the results, by eliminating biases. But it didn't show through the RMSE.

Trend:

1. As K increases, RMSE increases.
2. As computation becomes more complex, the running time increasing.

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.

1. I read in the data and put it into scipy sparse matrices, which made the computations a lot faster.
2. Before online prediction, I precompute all the L2-norms of the data points

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

I tried to do all of the computations by matrix operations instead of looping, which made the system run fast. So, the relevant data structure is as follows:

1. Scipy sparse matrix to represent user-movie matrix, centroid matrix during k-means.
2. Numpy matrix to represent computed models.
3. Numpy array to represent single data points, similarity vectors, and data point to cluster dictionaries.

Tools: JetBrains Pycharm 4.5.4

Libraries: Numpy, Scipy

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

Strengths:

1. Running very fast.
2. Can obtain good results with relatively low RMSE.
3. Highly modularized, reusable, and extensible.

Problem:

Bipartite clustering doesn't seem to improve the results.