Your Name: Leonardo Neves

Your Andrew ID: lneves

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 have written all the code for this assignment and the report all by myself.

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 and 90
Top 5 NNs of user 4321 in terms of cosine similarity	8497, 9873, 7700, 8202 and 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	1.1817295	35.65
Mean	Dot product	100	1.1769461	91.86
Mean	Dot product	500	1.1767977	342.33
Mean	Cosine	10	1.1817295	34.06
Mean	Cosine	100	1.1769507	96.40
Mean	Cosine	500	1.1768030	348.43
Weighted	Cosine	10	1.1825786	38.15
Weighted	Cosine	100	1.1781862	97.06
Weighted	Cosine	500	1.1794929	348.08

*runtime should be reported in seconds.

2.2 Movie-movie similarity

Rating Method	Similarity Metric	K	RMSE	Runtime(sec)
Mean	Dot product	10	1.179869	512.87
Mean	Dot product	100	1.169794	557.85
Mean	Dot product	500	1.170965	747.69
Mean	Cosine	10	1.179869	510.08
Mean	Cosine	100	1.169794	560.66
Mean	Cosine	500	1.170965	728.46
Weighted	Cosine	10	1.182782	510.62
Weighted	Cosine	100	1.197195	567.31
Weighted	Cosine	500	1.566751	747.37

2.3 Movie-rating/user-rating normalization

Rating Method	Similarity Metric	K	RMSE	Runtime(sec)
Mean	Dot product	10	1.179664	109.20
Mean	Dot product	100	1.177222	156.40
Mean	Dot product	500	1.177130	396.69
Mean	Cosine	10	1.179664	99.13
Mean	Cosine	100	1.177222	155.89
Mean	Cosine	500	1.177130	400.27
Weighted	Cosine	10	1.180433	100.73
Weighted	Cosine	100	1.178440	163.03
Weighted	Cosine	500	1.179814	410.45

Add a detailed description of your normalization algorithm.

For my normalization algorithm, I have selected a simple approach based on experiment 1. As the PCC gives a value from – 1 to 1, I have multiplied the user similarity ratings by the PCC of those two users so that users that have a negative or zero correlation would be ranked lower on the KNN algorithm. This way, my top KNN would be populated mostly with high correlation users, expecting my results to be more reliable.

2.4 Matrix Factorization

a. Briefly outline your optimization algorithm for PMF

I have selected the gradient descent algorithm for the optimization part. I took the derivative of the PMF function in respect to V and U and wrote the update rule for V as V' = V + alpha*(U.T*R - beta*lambda_v*V) and a similar rule for U. I have first fixed V and updated U

until it got stable and then started updating V fixing U. I repeated this process until reaching stopping criteria. V and U were sampled from a Gaussian distribution with 0 mean. My alpha would be changed during the optimization part, making it smaller if the step was big enough to make the error larger.
To make sure my predictions have been on the range [1-5], I have used the sigmoid function to make my ratings to be from 0 to 1 and them changed the range to 1-5.
For the matrix factorization, I removed the imputation from the preprocessing step, having my training predictions to be between 1-5.

b. Describe your stopping criterion

My algorithm would stop when the error changes between two iterations is smaller than 0.1% of the previous error.

Num of Latent Factors	**RMSE**	**Runtime(sec)***	**Num of Iterations**
2	1.12101384118	345.35	42
5	1.11948492795	375.52	27
10	1.12087255322	534.032	30
20	1.11910995753	1316.21	28
50	1.12229828956	2290.93	21

*runtime should be reported in seconds.

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.

I think the KNN, under the given assumptions, worked as expected. Because we would assume no ratings as a 3.0 rating and sort our nearest neighbors based on the most similar users and movies, it was expected that the predictions would be around 3.0. A better result would come from selecting only the kNN that had rated the movie, making the results more accurate. My PMF got a better result, close to the paper baseline but using much less data then the used by their algorithm. Also, they have used 30 latent factors, which I assume would be around the best value for it since my best results came from 20. Also. PMF is much slower than kNN for a smaller improvement. Depending on the context, it would have been better to use kNN in order to get good enough result fast enough.

One weird result from my PMF is having the 10 factors RMSE to be larger than my 5 factors RMSE. This might happen because of the randomness of the algorithm initialization. Since the results are pretty similar, a good initialization for 5 and a bad initialization for 10 might have resulted on this behavior.

My normalization approach was worse than expected. I would guess it would be because of sparseness (maybe using it with the movie rating might have been better since the algorithm from experiment 2 works better with sparse data) . My most correlated users might not have rated the movies, making my results worse. Despite of that, I still think it was a reasonable result/

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.

To preprocess the data, I have created an inverted list of users to the respective movie#ratings. This way, each line of my new file would be easily turned into a matrix of shape #users x #movies with the rating as the elements.

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

I have used sparse matrices from scipy to store the ratings and make computation faster. I have also used scipy to compute the sigmoid function for the PMF and sklearn to compute the cosine similarity. Numpy was used to sample the U and V matrices from a zero mean Gaussian.

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

My design was really fast for most the experiments and I believe having achieved the expected accuracy. One problem came up on the last experiment, as gradient descent as implemented was taking a long time to converge.