Data Mining Assignment 1 Report

CS21MDS14009

Srijit Sen

Instructions:

- The code was created in Python on personal laptop. Runtime may vary system to system
- Replace the csv path locations at line no 17 and 18
- Install all libraries before running the code. All library imports are done at very start of the code

Question 1:

Tags data was directly used to get counts of question tagged to each tag id.
For answer, Posts data was used with a self-join of answer id with question id

User Id	Answer Count	Rank
15811	4036	1
158442	2738	2
66509	2624	3

Tag Id	Question Count	Rank
175	4036	1
538	2738	2
140	2624	3

Question 2:

All User ids and Tags which were answered and annotated less than 20 times were removed
With the remaining, Tags were converted into individual tag columns with count of times annotated with a question for user ids to create the expert matrix

The Dimension of the expert matrix"
(2355, 2099) Including user id column otherwise (2355, 2098)

Question 3:

The count was converted to rating using the formula provided in the question to create the utility matrix.
Utility matrix was split into Test by extracting the bottom and left most 15% of the utility matrix
And train data was created from utility matrix directly but initializing all test cases inside train as 0

Utility matrix:

- Summation value of the utility matrix : 822,257
- Highest row sum of the utility matrix: 15811 (User id)
- Highest column sum of the utility matrix: 140 (Tag id)

Train and test data

- Summation value of the train matrix: 80,380
- Dimension of the test matrix: (354,235)
- Summation value of test matrix: 1877

Question 4:

Three user defined functions were created

- User-User prediction
- Item-Item prediction
- RMSE Calculation

For training and prediction, KNN is used to calculated nearest neighbors then using a loop for each user or item the neighbors are picked up and the rating is predicted by either taking a average of ratings of the neighbors or by taking weighted average. Predicted values are returned and passed to the RMSE function to calculate the accuracy. The RMSE is only calculated for test case rating >0

Method	Rating Prediction Function	Metric	N=2	N=3	N=5
Item-Item	Simple average	RMSE	2.2272685	2.215652	2.105392
	Weighted average	RMSE	2.2272685	2.215646	2.105692
User-User	Simple average	RMSE	2.2027173	2.155755	2.070477
	Weighted average	RMSE	2.2027173	2.155620	2.070903

The RMSE is unimpressive for all combinations, but it tends to decrease as neighbors increases

Question 5:

2 user defined functions were created

- matrix_factorization
- RMSE Calculation

For training and prediction 3 loop code is created to traverse through the matrix and pass it to gradient descent algorithm for multiple epochs. Predicted values are returned and passed to the RMSE function to calculate the accuracy. The RMSE is only calculated for test case rating >0

Method	Metric	K=2	K=5	K=10
Without Regularization	RMSE	1.335526	1.593951	1.755219
With Regularization $\lambda_1 = 0.001, \lambda_2 = 0.003$	RMSE	1.372190	1.602625	1.661273
$\lambda_1 = 0.05, \lambda_2 = 0.05$	RMSE	1.270182	1.274541	1.317849
$\lambda_1 = 0.50, \lambda_2 = 0.75$	RMSE	1.258652	1.251955	1.252905

The RMSE tends to go down with higher regularization values and higher latent factors

Question 6:

3 user defined functions were created

- Surprise Prediction for KNN baseline implementation
- SVD Implementation
- RMSE Calculation

For training and prediction, a new train and test dataset had to be created by melting the utility matrix into three column data – user id, tag id and ratings. For KNN Baseline, Cross validation is used to find the best algo parameters with split of 10 validation datasets. Best algorithm is then used to predict the test data

Predicted values are returned and passed to the RMSE function to calculate the accuracy. The RMSE is only calculated for test case rating >0

Algorithm	Method	RMSE for N=2	RMSE for N=3	RMSE for N=5
Item-Item	Your method	2.2272685	2.215646	2.105692
	Surprise	1.3237088	1.304686	1.322879
User-User	Your method	2.2027173	2.155620	2.070903
	Surprise	1.4997100	1.471392	1.494432

Surprise Method made better predictions than functions created from scratch

Method	RMSE for K=2	RMSE for K=5	RMSE for K=10
Your method	1.258652	1.251955	1.252905
Surprise	1.246306	1.236457	1.230257

Surprise Method made slightly better predictions than functions created from scratch, but the difference was not that much