CS550: Massive Data Mining and Learning Spring 2022

Problem Set 1

Due 11:59pm Monday, February 21, 2022
Please see the homework file for late policy

Honor Code: Students may have discussions about the homework with peers. However, each student must write down their solutions independently to show they understand the solution well enough in order to reconstruct it by themselves. Students should clearly mention the names of all the other students with whom they have discussions about the homework. Directly using the code or solutions obtained from the web or from others is considered an honor code violation. We check all the submissions for plagiarism and take the honor code seriously, and we hope students to do the same.

Discussions (People with whom you discussed ideas used in your answers):

On-line or hardcopy documents used as part of your answers:

I acknowledge and accept the Honor Code.

(Signed)_____Vaishnavi Manthena_____

If you are not printing this document out, please type your initials above.

Answer to Questions 1

Description of Algorithm Used

At a high level I used 3 Map Reduce Jobs:

Map-Reduce Job 1

Map

- Each input line is converted to several key & value pairs.
- Say an input line has 'N' friends. The ith key-value pair generated from this input line would be as follows.
- Key: id of friend i
- Value: (user id, friend1, friend2, ….., friendN). This is basically the user information including user id and a list of all friends.
- Every key-value pair generated from a single line has the same value.

Reduce

- Key: id of friend
- Value: List of user information.
- So, this tells us about all the users who are friends with the key. A pair of users in this list could potentially have the key as a mutual friend. They won't in case they are friends with each other in which case one user would be in the friends list of the other. The next map process basically uses such information from each value in this reduce result to generate pairs of users who are not friends with each other but have the mutual friend represented by the key.

Map-Reduce Job 2

Map

- Key: (User 'i', User 'j')
- Value: 1
- Meaning of this key, value pair: User 'i' and User 'j' appeared in a same value of previous reduce result. Also, they are not friends with eachother.
- The total number of times (User i, User j) is outputted from this map program is equal to the total number of mutual friends that users 'i' and 'j' have.
- This mapping program ensures that every time a (key, value) pair (i,j) is produced, then (j,i) is also produced. This helps in the next map reduce step to collect the recommendations for each user.

Reduce

- Key: (User 'i', User 'j')
- Value: total number of mutual friends (similarity between users 'i' and 'j').

Map-Reduce Job 3

Map

- Main goal: To convert each (key, value) pair of the previous reduce job into another format.

- Key: User i
- Value: User j, similarity
- Meaning. Same as before. User 'i' and User 'j' are not friends with each other but have 'similarity' number of mutual friends.

Reduce

- Key: User i
- Value: List of (user j, similarity pairs)

I used several spark transformations to achieve the above main map-reduce jobs.
After these map-reduce jobs, I used simple transformations to perform functions to give required output. For example, to sort the values appropriately, finalize their formats, and also sort the keys.

Note: According to my algorithm, all the users you don't have any friends are given an empty list of recommendations and these users are listed together at the end of the output.

I did this question using spark. My result is a java project, which is built with maven using a pom.xml file. So, the whole program is given in the project-template folder. My actual implementation is in Question_1.java which is there as you go into the folders one by one. Apart from this I used only one other source file (Point.java) in the same location as Question_1.java. I fit all my main logic in Question_1.java itself, but due to time constraints I used Point.java only for a minor task towards the end to sort the tuples in descending order of similarity and ascending order of id.

I ran my program on iLab1.cs.rutgers.edu.
Instructions to run my program:

1. **cd into the project-template directory**
2. **execute the command "mvn install"**
3. **execute the command "mvn clean package"**
4. **execute command:**
   spark-submit target/Question_1.jar path_to_input_file

Output for certain user id's:
924 439,2409,6995,11860,15416,43748,45881
8941 8943,8944,8940
8942 8939,8940,8943,8944
9019 9022,317,9023
9020 9021,9016,9017,9022,317,9023
9021 9020,9016,9017,9022,317,9023
9022 9019,9020,9021,317,9016,9017,9023
9990 13134,13478,13877,34299,34485,34642,37941
9992 9987,9989,35667,9991
9993 9991,13134,13478,13877,34299,34485,34642,37941

Answer to Questions 2(a)

Ignoring Pr(B) is a drawback because it allows a rule to have high confidence only because 'B' is a very common item/itemset. For instance, if 'B' occurs in every basket then the conf (A -> B) would be 1, although it is not true that A and B are bought together very frequently. It is just that B is bought frequently. Hence, Ignoring Pr(B) may lead to rules that are less interesting.

To calculate lift, the conf(A->B) is divided by Pr(B) or equivalently S(B) = Support(B)/N. Therefore, the higher the Pr(B) value, the less interesting the association rule is, and the lower the lift value. The lower the Pr(B) value, the more interesting the rule is, and the higher the lift value.

Similarly, conv(A->B) uses 1-S(B) as the numerator. Which means that conviction increases and decreases with decreasing and increasing values of S(B) respectively.

So, lift and conviction appropriately incorporate Pr(B) and take interestingness of a rule into consideration.

Answer to Questions 2(b)

Confidence is not symmetrical.

$$conf(A \rightarrow B) = \frac{P(A\ and\ B)}{P(A)}$$

$$conf(B \rightarrow A) = \frac{P(A\ and\ B)}{P(B)}$$

Clearly not symmetrical cause numerators are equal but denominators are not.

Counterexample:
Suppose there are 8 baskets.
5 of them have item 'm.'
6 of them have item 'b.'
4 of them have itemset {m, b}.

$$conf(b \rightarrow m) = \frac{P(b\ and\ m)}{P(b)} = \frac{4}{6}$$

$$conf(m \rightarrow b) = \frac{P(b\ and\ m)}{P(m)} = \frac{4}{5} \neq \frac{4}{6}$$

$$conf(b \rightarrow m) \neq conf(m \rightarrow b).$$

Lift is symmetrical.

$$\boldsymbol{lift}(\boldsymbol{A \rightarrow B}) = \frac{conf(A \rightarrow B)}{S(B)} = \frac{\frac{P(A\ and\ B)}{P(A)}}{S(B)} = \frac{\frac{S(A\ and\ B)}{S(A)}}{S(B)} =$$

$$\frac{\frac{\frac{Support(A\ and\ B)}{N}}{\frac{Support(A)}{N}}}{\frac{Support(B)}{N}} = \frac{\frac{Support(A\ and\ B)}{Support(A)}}{\frac{Support(B)}{N}} = \frac{\boldsymbol{Support(A\ and\ B)}}{\boldsymbol{Support(A)}} * \frac{\boldsymbol{N}}{\boldsymbol{Support(B)}}$$

$$\boldsymbol{lift}(\boldsymbol{B \rightarrow A}) = \frac{conf(B \rightarrow A)}{S(A)} = \frac{\frac{P(A\ and\ B)}{P(B)}}{S(A)} = \frac{\frac{S(A\ and\ B)}{S(B)}}{S(A)} =$$

$$\frac{\frac{\frac{Support(A\ and\ B)}{N}}{\frac{Support(B)}{N}}}{\frac{Support(A)}{N}} = \frac{\frac{Support(A\ and\ B)}{Support(B)}}{\frac{Support(A)}{N}} = \frac{\boldsymbol{Support(A\ and\ B)}}{\boldsymbol{Support(B)}} * \frac{\boldsymbol{N}}{\boldsymbol{Support(A)}}$$

Therefore, $\boldsymbol{lift}(\boldsymbol{A \rightarrow B}) = \boldsymbol{lift}(\boldsymbol{B \rightarrow A})$.

Conviction is not symmetrical.

Using the same counter-example as in confidence.

$$conv(b \rightarrow m) = \frac{1 - S(m)}{1 - conf(b \rightarrow m)} = \frac{1 - 5/8}{1 - 4/6} = \frac{3/8}{2/6} = \frac{9}{8}$$

$$conv(m \rightarrow b) = \frac{1 - S(b)}{1 - conf(m \rightarrow b)} = \frac{1 - 6/8}{1 - 4/5} = \frac{2/8}{1/5} = \frac{5}{4}$$

$$conv(b \rightarrow m) \neq conv(m \rightarrow b)$$

Answer to Questions 2(c)

Confidence holds this property. Since, it is a probability, its maximal value is 1. If '$A \rightarrow B$' is a perfect implication, then its confidence would be 1. Since 'B' would occur in every basket that has 'A', $\frac{P(A\ and\ B)}{P(A)} = 1$.

Lift does not have this property. It is quite possible that perfect implications have a lower lift value than imperfect ones.

$$lift(A \rightarrow B) = \frac{conf(A \rightarrow B)}{S(B)}$$

Suppose there is a perfect implication 'A -> B' for which S(B) = 0.4.
Suppose there is an imperfect implication 'C->D' whose confidence is 0.8 and S(D) = 0.3.

Lift (A->B) = 1/0.4 = 2.5
Lift (C->D) = 0.8/0.3 = 2.67

So, lift's measure is not maximal for perfect implications.

Conviction has this property.

$$conv(A \rightarrow B) = \frac{1 - S(B)}{1 - conf(A \rightarrow B)}$$

For perfect implication's (confidence is 1), the denominator becomes 0 and its value reaches ∞.

Implementation of the Apriori Algorithm is in Question_2.py.

Answer to Questions 2(d)

Top 5 rules (from frequent pairs) and corresponding confidence scores:
DAI93865->FRO40251: 1.0
GRO85051->FRO40251: 0.999176276771005
GRO38636->FRO40251: 0.9906542056074766
ELE12951->FRO40251: 0.9905660377358491
DAI88079->FRO40251: 0.9867256637168141

Answer to Questions 2(e)
Top 5 rules (from frequent triples) and corresponding confidence scores:
(DAI23334, ELE92920)->DAI62779: 1.0
(DAI31081, GRO85051)->FRO40251: 1.0
(DAI55911, GRO85051)->FRO40251: 1.0
(DAI62779, DAI88079)->FRO40251: 1.0
(DAI75645, GRO85051)->FRO40251: 1.0

Answer to Questions 3(a)

$$P(don't\ know) = P(all\ k\ rows\ have\ 0's)$$
$$= \frac{total\ number\ of\ ways\ to\ choose\ k\ rows\ from\ the\ n-m\ 0\ rows}{total\ number\ of\ ways\ to\ choose\ k\ rows\ from\ n\ rows}$$

$$= \frac{(n-m)C(k)}{nCk} = \frac{\frac{(n-m)!}{(n-m-k)!(k)!}}{\frac{n!}{(n-k)!k!}} = \frac{(n-m)!(n-k)!}{(n-m-k)!n!} = \frac{(n-m)!}{n!} * \frac{(n-k)!}{(n-m-k)!}$$

$$= \frac{1}{n(n-1)(n-2)\ldots.(n-m+1)} * \frac{(n-k)(n-k-1)\ldots(n-k-m+1)}{1}$$

$$= \frac{(n-k)(n-k-1)\ldots(n-k-m+1)}{n(n-1)(n-2)\ldots.(n-m+1)} \leq \frac{(n-k)(n-k)\ldots..(n-k)}{n*n*n*\ldots\ldots*n} = \left(\frac{n-k}{n}\right)^m$$

In the above line we were able to go from term 1 to term 2 because when b>a and b>= 2 : $\frac{a}{b} > \frac{a-1}{b-1}$. This is because $\frac{a}{b} - \frac{a-1}{b-1} = \frac{b-a}{b(b-1)}$. This is positive for the mentioned constraints on 'a' and 'b'.

Also, we were able to go from term 2 to term 3 because there are 'm' factors in the numerator and denominator.

Answer to Questions 3(b)

To get the smallest value of 'k' that will assure that the probability of "don't know" is at most e^{-10}, the below equation needs to be solved for k.

$$\left(\frac{n-k}{n}\right)^m = \left(\frac{1}{e}\right)^{10}$$

Simplifying the left-hand side:

$$\left(\frac{n-k}{n}\right)^m = \left(1-\frac{k}{n}\right)^m = \left(1-\frac{1}{n/k}\right)^{\frac{n}{k}*\frac{k}{n}*m} = \left(\left(1-\frac{1}{\frac{n}{k}}\right)^{\frac{n}{k}}\right)^{\frac{k}{n}*m}$$

Since 'n' is much larger than 'k,' n/k is a very large number.

So, $\left(1-\frac{1}{\frac{n}{k}}\right)^{\frac{n}{k}}$ can be approximated to 1/e.

Therefore, the initial equation simplifies to:

$$\left(\frac{1}{e}\right)^{\frac{k}{n}*m} = \left(\frac{1}{e}\right)^{10}$$

Solving for k:

$$\frac{k}{n} * m = 10$$

$$\boldsymbol{k = \frac{10 * n}{m}}$$

Answer to Questions 3(c)

PERMUTATION 1	PERMUTATION 2	PERMUTATION 3	PERMUTATION 4	S1	S2
1	2	3	4	0	0
2	3	4	1	1	1
3	4	1	2	1	0
4	1	2	3	0	0

$$\text{Jaccard Similarity}(S1, S2) = \frac{size\ of\ intersection}{size\ of\ union} = \frac{1}{2}$$

	S1	S2
Min hash Value for Permutation 1	2	2
Min hash Value for Permutation 2	3	3
Min hash Value for Permutation 3	1	4
Min hash Value for Permutation 4	1	1

$$P(a\ random\ cyclic\ permutation\ yields\ the\ same\ minhash\ value\ for\ both\ S1\ and\ S2)$$

$$= \frac{\#\ of\ cyclic\ permutation\ for\ which\ S1\ and\ S2\ have\ same\ minhash\ value}{Total\ \#\ of\ possible\ cyclic\ permutations}$$

$$= \frac{3}{4}$$

$$\frac{1}{2} \neq \frac{3}{4}$$