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:

Lecture slides

I acknowledge and accept the Honor Code.

KK____

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

Answer to Question 1:

(i) File submitted: hw_1_q1.py

(ii) Description of the algorithm:

First, find the friends and mutual friends of all users. If two people are friends, then put 0, else if two people are mutual friends, put 1 for the "value" in the key-value pair. We can do this in this way:

Mapper:
Input: userID1 -> userID2 userID3 userID4
Output:
friends_list: [((userID1,userID2),0),((userID1,userID3),0),((userID1,userID4),0)]
mutual_friends_list: [((userID2,userID3),1), ((userID2,userID4),1), ((userID3,userID2),1), ((userID3,userID4),1), ((userID4,userID2),1), ((userID4,userID3),1)]

Now we return the friends_list+mutual_friends_list as relationship_list to reducer.

Reducer:
In the reducer, we filter out the key-value pairs whose value is 0 (meaning, the two users are already friends) from relationship_list using the subtractByKey() method.

Now reduceByKey() is used to group the items by their key values. Now the data is like:
key: (userID1,userID2) ; value: num_of_mutual_friends
[((user1,user2),10), ((user3,user5),5), ((user1,user4),8)]

Using map() function, we change the data to look like this:
key: userID ; value: (non_friend_user, num_of_mutual_friends)
[(user1,(user2,10)), (user3,(user5,5)), (user1,(user4,8))]

We apply groupByKey().mapValues(list) to convert the data into this form:
key: userID value: [(non_friend_user1, num_of_mutual_friends),(non_friend_user2, num_of_mutual_friends) …]
[(user1, [(user2,10),(user4,8)]), (user3, [(user5,5)])]

Now we can recommend the top 10 friends by sorting the array for each person/user in reverse order of highest mutual friends, with tie breaker of ascending order of user IDs.

(iii) Recommendations:

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 Question 2:

Answer to (2a)

Pr(B) represents the probability of finding the item set B in the basket. This may lead to incorrect results. Confidence calculates the probability of B being purchased if A was purchased.If the occurrence of item set B is independent of item set A, i.e., Pr(B|A) = Pr(B) but support(B) is high, then Confidence(A->B) gives a high value implying and A->B is identified as a valid rule, but not for the right reason. Conviction and Lift take support of B into account, so they do not suffer from this drawback.

Answer to (2b)

Symmetrical measures(s):

Lift

Proof:

Lift(A -> B) = $\frac{conf(A->B)}{S(B)}$ = $\frac{Pr(B \cap A)}{Pr(A)\frac{\#transactions\ of\ B}{N}}$ = $\frac{Pr(B \cap A)}{Pr(A)Pr(B)}$

Lift(B -> A) = $\frac{conf(B->A)}{S(A)}$ = $\frac{Pr(A \cap B)}{Pr(B)\frac{\#transactions\ of\ A}{N}}$ = $\frac{Pr(A \cap B)}{Pr(B)Pr(A)}$

Asymmetrical measures:

Confidence

Proof:

conf(A->B) = Pr(B|A) = $\frac{Pr(B \cap A)}{Pr(A)}$

conf(B->A) = Pr(A|B) = $\frac{Pr(A \cap B)}{Pr(B)}$

Clearly Pr(A) need not be the same as Pr(B). So confidence is asymmetrical

Conviction

Proof:

We already know conf(A -> B) ≠ conf(B -> A) => denominators are not equal in both directions
And S(A) ≠ S(B) => numerators are not equal in both directions

So conv(A -> B) ≠ conv(B -> A) because $\frac{1-S(B)}{1-conf(A->B)} \neq \frac{1-S(A)}{1-conf(B->A)}$

Counter example:

Baskets:
{A,B} ,{B,D} , {D,A} , {A,C}
S(A) = ¾
S(B) = 2/4
Pr(A∩B) = ¼
conf(A -> B) = $\frac{1/4}{3/4}$ = 1/3
conf(B -> A) = $\frac{1/4}{2/4}$ = 1/2
So they are not equal

Similarly,
conv(A->B) = $\frac{1-2/4}{1-1/3}$ = 3/4
conv(B -> A) = $\frac{1-3/4}{1-1/2}$ = ½
They are not same as well.

Answer to (2c)

Let's take baskets:
{A,B},{C,D},{C,D},{E,F}

We know Pr(D|C) = 1, Pr(F|E) = 1
S(D) = 2/4 = 1/2
S(F) = ¼

conf(C->D) = Pr(D|C) = 1
conf(E->F) = Pr(F|E) = 1

So the value of conf is maximal for rules that hold 100% of the time.

conv(C->D) = $\frac{1-1/2}{1-1}$ = ∞
conv(E->F) = $\frac{1-1/4}{1-1}$ = ∞

So the value of conv is maximal for rules that hold 100% of the time.

lift(C->D) = $\frac{1}{1/2} = 2$
lift(E->F) = $\frac{1}{1/4} = 4$

The value of lift is different for the rules that hold 100% of the time.

In conclusion, conf, conv are desirable while lift is not.

Answer to (2d)

Top 5 pairs with support = 100

```
DAI93865  =>  FRO40251   1.0
GRO85051  =>  FRO40251   0.999176276771005
GRO38636  =>  FRO40251   0.9906542056074766
ELE12951  =>  FRO40251   0.9905660377358491
DAI88079  =>  FRO40251   0.9867256637168141
```

Answer to 2(e)

Top 5 triplets with support = 100

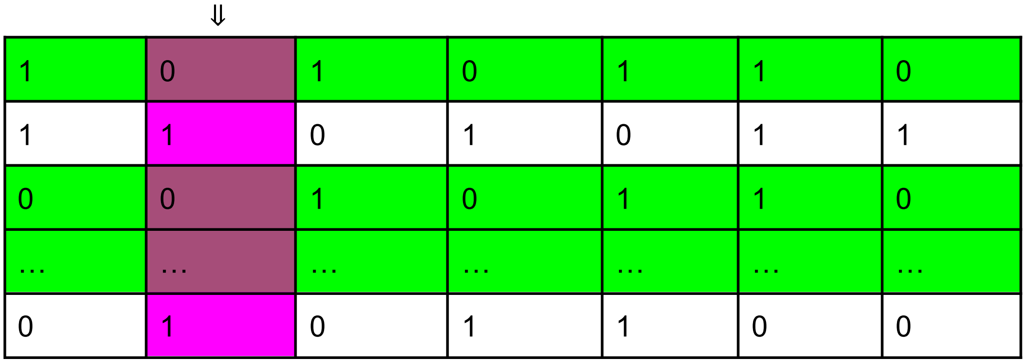
```
DAI23334  ,  ELE92920  =>  DAI62779   1.0
DAI31081  ,  GRO85051  =>  FRO40251   1.0
DAI55911  ,  GRO85051  =>  FRO40251   1.0
DAI62779  ,  DAI88079  =>  FRO40251   1.0
DAI75645  ,  GRO85051  =>  FRO40251   1.0
```

Answer to Question 3:

Answer to (3a)

There are n rows and we randomly choose k rows from them. Out of these chosen rows, we consider a particular column and see if there is at least one 1 in that column. If there are no 1s in that column, then the result of min hashing is 'don't know'.

The diagram is like this:
Let's say the k rows are the green rows, we are considering the 2nd column.



1	0	1	0	1	1	0
1	1	0	1	0	1	1
0	0	1	0	1	1	0
…	…	…	…	…	…	…
0	1	0	1	1	0	0

The pink column is what we selected, the random k rows are green rows, so the intersection, i.e., magenta blocks are what we consider finally. If all these magenta blocks have 0s, then the resulting min hashing is "don't know".

Let's calculate the probability of getting all 0s in the magenta blocks.

M rows contain 1 in the selected column, n-m rows contain 0s.
So the number of ways to select k rows from (n-m) rows so that there are all 0s in the k rows is:
${}^{(n-m)}C_k = \frac{(n-m)!}{(n-m-k)!k!}$ – (eq1)

Number of ways to choose k rows from n rows: ${}^{n}C_k = \frac{n!}{(n-k)!k!}$ – (eq2)

Probability of getting "don't know" = $\frac{(eq1)}{(eq2)}$

=> $\frac{(n-m)!(n-k)!k!}{(n-m-k)!k!n!}$

=>$\left(\frac{n-k}{n}\right)\left(\frac{n-k-1}{n-1}\right)\ldots\left(\frac{n-k-(m-1)}{n-(m-1)}\right)\left(\frac{(n-k-m)!(n-m)!}{(n-m)!(n-m-k)!}\right)$

=>$\left(\frac{n-k}{n}\right)\left(\frac{n-k-1}{n-1}\right)\ldots\left(\frac{n-k-(m-1)}{n-(m-1)}\right)$

Each term in the above equation is $\leq \frac{n-k}{n}$ and there are m terms.

If we replace all terms with the maximum possible value of $\frac{n-k}{n}$, we get the at most probability as $\left(\frac{n-k}{n}\right)^m$

Hence proved.

Answer to (3b)

We know that the probability of "don't know" = $\left(\frac{n-k}{n}\right)^m$

And in the question, it is given that $n \gg m, k$

We want to calculate the approximation to the smallest value of k that will assure this probability is at most e^{-10}

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

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

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

In the question it is given that for large x, $\left(1 - \frac{1}{x}\right)^{x} \approx 1/e$ we know that $n \gg k$ => n/k is a large value.

So we use this and get: $(1/e)^{mk/n} \leq e^{-10}$

=> $e^{-mk/n} \leq e^{-10}$ => $\frac{-mk}{n} \leq -10$ => $k \geq \frac{10n}{m}$

So the smallest value of k is $\frac{10n}{m}$

Answer to (3c)

Let the two sets be:

S1
1
0
1
0

S2
0
0
1
1

Jaccard Similarity: $\frac{S1 \cap S2}{S1 \cup S2}$ = $\frac{1}{3}$

C1	C2	C3	C4
1	4	3	2
2	1	4	3
3	2	1	4
4	3	2	1

S1	S2
1	0
0	0
1	1
0	1

Signature Matrix	
1	3
2	2
1	1
2	1

From the table, the probability that a random cyclic permutation yields the same min-hash value for both S1 and S2 is $\frac{2}{4}$ = $\frac{1}{2}$

Hence the probability that the min-hash values agree over cyclic permutations is not the same as the Jaccard Similarity.