CS550: Massive Data Mining and Learning
Homework 1

Due 11:59pm Thursday, March 5, 2020
Only one late period is allowed for this homework (11:59pm Friday 3/6)

Submitted by:
Name: Prakruti Joshi
NetID: phj15

Submission Instructions

Assignment Submission Include a signed agreement to the Honor Code with this assignment. Assignments are due at 11:59pm. All students must submit their homework via Sakai. Students can typeset or scan their homework. Students also need to include their code in the final submission zip file. Put all the code for a single question into a single file.

Late Day Policy Each student will have a total of *two* free late days, and for each homework only one late day can be used. If a late day is used, the due date is 11:59pm on the next day.

Honor Code Students may discuss and work on homework problems in groups. This is encouraged. 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 who were part of their discussion group. Using code or solutions obtained from the web is considered an honor code violation. We check all the submissions for plagiarism. We take the honor code seriously and expect students to do the same.

Discussion Group (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) Prakruti Joshi

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

Answer to Question 1

1. Code attached in submission.
2. ***Algorithm:***

   Input data Type: Multiple lines of format: [UserID][Tab][Friends ID]
   Approach:
   Use a simple MapReduce job to process the data and generate recommendations.

   Mapper class:

   - Reads input data line by line. Stores each line in list by splitting on tab.
   - Input to mapper:
     Key: UserID [Type: LongWritable]
     Value: List of friends [Type: String array]
   - If friend list is not empty, then the map generates output pair as follows:
     Key: UserId of friend [Type: LongWritable]
     Value: Recommended Friend and mutual friend [Type: FriendsCountWritable]
   - *FriendsCountWritable* is customized writable data type which has two attributes:
     1. *userID* of type Long 2. *mutualFriends* of type Long
   - The mapper generates:

     $$< Friend1; Friend2, m = UserID >$$
     $$< Friend2; Friend1, m = UserID >$$

     also for all the combinations of friends of a particular user. Say a user has n friends, then the mapper generates n*(n-1) such pairs.
     $< Friend1; Friend2, m = UserID >$ means that Friend1 is recommended to Friend2 and has UserID as the mutual friend.
   - Mapper also generates $< UserID; Friend1, m = -1 >$ to indicate if the recommended friend and the user are already friends. By checking the value of *mutualFriends* = -1, we can verify this.
   - For example, if a User 1 has friends 2 and 3, following will be generated:
     $< 2; 3, 1 >, < 3; 2, 1 >, < 1; 2, -1 >, < 1; 3, -1 >$

   Reduce Class:

   - Input to Reduce class: $< Key, Value >=< LongWritable, FriendsCountWritable >$
   - Reduce uses a HashMap which stores the counts of number of mutual friends between the key and the recommended friend in the value of the input.
   - If the recommended friend is already a friend of user, then it is not included in the count. This is maintained by checking the value of *mutualFriends*.

- Reduce class uses *TreeMap* to sort the Hashmap based on descending number of mutual friends. In case of tie, it is sorts based on ascending order of User Ids.
- The output of Reduce is the user ID of each user and the list of recommended friend Ids:$< Key, Value >=< LongWritable, Text >$
- These recommendations for each user are printed as output.

3. ***Recommendations for the users with the following ids:***

- **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(a)

Confidence of association rule $(A \rightarrow B)$ is:

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

Confidence does not take the individual probability of B into account. Thus, all the rules with high confidence values might not be interesting. For example in the market analysis, the rule $(X \rightarrow milk)$ might have high confidence as milk is purchased frequently. X and milk might be independent, and X may have less frequency, but since milk is very frequent, their co-occurrence is high and thus confidence is high. Rules generated in such cases are not useful for analysis and might be misleading. For example,

	Milk	$\neg Milk$	
Beer	15	5	20
$\neg Beer$	75	5	80
	90	10	100

From the above table, confidence$(Milk \rightarrow Beer) = 0.75$. But Pr(Coffee) = 0.9 .
Also, confidence$(Milk \rightarrow \neg Beer) = 0.9375$
Thus, even if the confidence is high, the association rule $(Milk \rightarrow Beer)$ is misleading.

Interest is a measure which takes the difference between confidence and the support of B. This provides a better intuition for the association rule.
Both lift and conviction take Pr(B) or support(B) into account. Thus, they do not face this drawback.

Answer to Question 2(b)

1. **Confidence:** Not symmetrical

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

Thus, $conf(A \rightarrow B) \neq conf(B \rightarrow A)$

Example 1:
If Pr(A) = 0.4, Pr(B) = 0.3 and $Pr(A \cap B) = 0.2$, then:
$conf(A \rightarrow B) = 0.2/0.4 = 0.50$
$conf(B \rightarrow A) = 0.2/0.3 = 0.67$

2. **Lift:** Symmetrical

$$lift(A \rightarrow B) = lift(A \rightarrow B) = \frac{Pr(B \cap A)}{Pr(A)Pr(B)}$$

3. **Conviction:** Not symmetrical
Since conviction is based on confidence and confidence is not symmetrical, hence conviction is also not symmetrical.

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

Example 2: Taking the same values as for example 1 in confidence counterexample,
$conv(A \rightarrow B) = \frac{(1-0.3)}{(1-0.2)} = \frac{0.7}{0.8} = 0.875$
$conf(B \rightarrow A) = \frac{(1-0.4)}{(1-0.2)} = \frac{0.6}{0.8} = 0.75$
Thus, $conv(A \rightarrow B) \neq conv(B \rightarrow A)$

Answer to Question 2(c)

Conviction and confidence have the property of desirable i.e. the value is maximal for perfect implications. Lift is not desirable.

Explanation:

1. Confidence: Confidence can take maximum value as 1. If B occurs every time A occurs then $Pr(A \cap B) = Pr(A)$. Thus, confidence value is 1.

2. Conviction: If B occurs every time A occurs, then conviction has the value ∞ which is maximal. This is because $conf(A \rightarrow B) = 1$.

3. Lift:

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

   Thus, lift depends on Pr(B). If B occurs every time A occurs, the confidence will have value 1. But the value of lift will depend on Pr(B) and might not be maximal in every case.

   Example:
   If we have a list of the following baskets as input:
   $[AB, AB, ABE, CD, CDF, EF, F, ABF]$
   Then for the perfect association rules: $(A \rightarrow B), (C \rightarrow D)$:
   $conf(A \rightarrow B) = 1$ and $conf(C \rightarrow D) = 1$
   $conv(A \rightarrow B) = \infty$ and $conv(C \rightarrow D) = \infty$
   However,
   $lift(A \rightarrow B) = \frac{conf(A \rightarrow B)}{Pr(B)} = \frac{1}{0.5} = 2$
   $lift(C \rightarrow D) = \frac{conf(C \rightarrow D)}{Pr(C)} = \frac{1}{0.25} = 4$

   Thus, lift is not desirable for perfect implications.

Answer to Question 2(d)

Top 5 pairs and their confidence scores:

Association rule	Confidence score
DAI93865 → FRO40251	1.0
GRO85051 → FRO40251	0.999176276771005
GRO38636 → FRO40251	0.9906542056074766
ELE12951 → FRO40251	0.9905660377358491
DAI88079 → FRO40251	0.9867256637168141

Code Output:

```
Top 5 pairs and their confidence scores:
  A    ->    B       Confidence score
DAI93865 -> FRO40251 1.0
GRO85051 -> FRO40251 0.999176276771005
GRO38636 -> FRO40251 0.9906542056074766
ELE12951 -> FRO40251 0.9905660377358491
DAI88079 -> FRO40251 0.9867256637168141

------------------------------------------------------------
```

Figure 1:

Answer to Question 2(e)

Top 5 triples and their confidence scores:

Association rule	Confidence score
(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

Code Output:

```
------------------------------------------------------------

Top 5 triples and their confidence scores:
  (A,B)   ->   C     Confidence score
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
```

Figure 2:

Answer to Question 3(a)

The column with n rows has m 1's and n-m 0's.
The number of ways of selecting k rows from n rows whose value is 0 is: $\binom{n-m}{k}$
Also, the total possible ways of selecting k rows from n rows is $\binom{n}{k}$
Thus, the probability of selecting k rows with value as 0 i.e. the probability of getting "don't know" as the min-hash value is: $\binom{n-m}{k}/\binom{n}{k}$ Now,

$$\begin{aligned}\frac{\binom{n-m}{k}}{\binom{n}{k}} &= \frac{\frac{(n-m)!}{k!(n-m-k)!}}{\frac{n!}{k!(n-k)!}} \\ &= \frac{(n-m)!(n-k)!}{n!(n-m-k)!} \\ = (\frac{n-k}{n})(\frac{n-k-1}{n-1})....(\frac{n-k-m+1}{n-m+1})\end{aligned}$$

Each term in the above product is approximately equal to $\frac{n-k}{n}$ and there are m terms in the product. Thus,

$$\frac{\binom{n-m}{k}}{\binom{n}{k}} \approx (\frac{n-k}{n})^m$$

Hence, the probability of getting "don't know" as the min-hash value is at most $(\frac{n-k}{n})^m$.

Answer to Question 3(b)

Goal: To find approximate value for smallest value of k such that the probability of "don't know" is at most e^{-10}.
Assumption: n is much larger than m and k

Let $(\frac{n-k}{n})^m$ be the probability of getting "don't know" as the min-hash value. Now we want this probability to be atmost e^{-10}. Thus,

$$\begin{aligned} (\frac{n-k}{n})^m &\leq e^{-10} \\ \implies (1-\frac{k}{n})^m &\leq e^{-10} \\ \implies ((1-\frac{k}{n})^{\frac{n}{k}})^{\frac{mk}{n}} &\leq e^{-10} \end{aligned}$$

Now k << n and n is large. Thus $((1-\frac{k}{n})^{\frac{n}{k}})$ can be approximated to $\frac{1}{e}$ using the property: For large x, $(1-\frac{1}{x})^x \approx \frac{1}{e}$. Thus,

$$\begin{aligned} (\frac{1}{e})^{\frac{mk}{n}} &\leq (\frac{1}{e})^{-10} \\ \implies \frac{mk}{n} &\geq 10 \\ \implies k &\geq \frac{10m}{n} \end{aligned}$$

Thus, the smallest approximate value for k is $\frac{10m}{n}$.

Answer to Question 3(c)

Example:

$$S1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad S2 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Jaccard similarity of S1 and S2 is 0.5. If we take normal min-hash values over normal permutations, i.e. min-hash values from any of the 3! permutations, we get that the min-hash values agree in half of the permutations. For example taking the 3! = 6 permutations of S1 and S2 we get:

Permutation	Min-hash Agree?
1 2 3	Yes
1 3 2	No
2 3 1	Yes
2 1 3	Yes
3 2 1	No
3 1 2	No

The min-hash values agree 50 percent of the times which is the same as Jaccard similarity. However, if we take cyclic permutations starting from last row of S1 and S2, the min-hash values differ, If we take cyclic permutations starting from either of the first two rows, then the min-hash values match. Thus, the probability of min-hash values matching is $\frac{2}{3}$. This is not the same as Jaccard similarity. Thus, the cyclic permutations are not sufficient to estimate Jaccard similarity.