CS246: Mining Massive Data Sets **Winter 2016**

Problem Set 1

Due 11:59pm January 21, 2016

Only one late period is allowed for this homework (11:59pm 1/26).

General Instructions

Submission instructions: These questions require thought but do not require long answers. Please be as concise as possible. You should submit your answers as a writeup in PDF format via GradeScope and code via the Snap submission site.

Submitting writeup: Prepare answers to the homework questions into a single PDF file and submit it via `http://gradescope.com`. Make sure that the answer to each question is on a *separate page*. This means you should submit a 15-page PDF (1 page for the cover sheet, 1 page for the answers to question 1, 5 pages for answers to question 2, 3 pages for question 3, and 5 pages for question 4). On top of each page write the number of the question you are answering. Please find the cover sheet and the recommended templates located here:
`http://web.stanford.edu/class/cs246/homeworks/hw1/submission_template_hw1.tex`
`http://web.stanford.edu/class/cs246/homeworks/hw1/submission_template_hw1.pdf`
`http://web.stanford.edu/class/cs246/homeworks/hw1/submission_template_hw1.docx`

Not including the cover sheet in your submission will result in a 2-point penalty.

Submitting code: Upload your code at `http://snap.stanford.edu/submit`. Put all the code for a single question into a single file and upload it.

Questions

1 MapReduce (25 pts) [Jeff/Sameep/Ivaylo]

Write a MapReduce program in Hadoop that implements a simple "People You Might Know" social network friendship recommendation algorithm. The key idea is that if two people have a lot of mutual friends, then the system should recommend that they connect with each other.

Input:
Download the input file from the link: `http://snap.stanford.edu/class/cs246-data/hw1q1.zip`.
The input file contains the adjacency list and has multiple lines in the following format:

`<User><TAB><Friends>`

Here, `<User>` is a unique integer ID corresponding to a unique user and `<Friends>` is a comma-separated list of unique IDs corresponding to the friends of the user with the unique ID `<User>`. Note that the friendships are mutual (i.e., edges are undirected): if A is friend with B then B is also friend with A. The data provided is consistent with that rule as there is an explicit entry for each side of each edge.

Algorithm: Let us use a simple algorithm such that, for each user U, the algorithm recommends $N = 10$ users who are not already friends with U, but have the largest number of mutual friends in common with U.

Output: The output should contain one line per user in the following format:

`<User><TAB><Recommendations>`

where `<User>` is a unique ID corresponding to a user and `<Recommendations>` is a comma-separated list of unique IDs corresponding to the algorithm's recommendation of people that `<User>` might know, ordered by decreasing number of mutual friends. Even if a user has fewer than 10 second-degree friends, output all of them in decreasing order of the number of mutual friends. If a user has no friends, you can provide an empty list of recommendations. If there are multiple users with the same number of mutual friends, ties are broken by ordering them in a numerically ascending order of their user IDs.

Also, please provide a description of how you are going to use MapReduce jobs to solve this problem. Don't write more than 3 to 4 sentences for this: we only want a very high-level description of your strategy to tackle this problem.

Note: It is possible to solve this question with a single MapReduce job. But if your solution requires multiple map reduce jobs, then that's fine too.

What to submit

(i) Submit the source code via the snap electronic submission website.

(ii) Include in your writeup a short paragraph describing your algorithm to tackle this problem.

(iii) Include in your writeup the recommendations for the users with following user IDs: 924, 8941, 8942, 9019, 9020, 9021, 9022, 9990, 9992, 9993.

2 Association Rules (30 pts) [Duyun/Nihit/Shubham]

Association Rules are frequently used for Market Basket Analysis (MBA) by retailers to understand the purchase behavior of their customers. This information can be then used for many different purposes such as cross-selling and up-selling of products, sales promotions, loyalty programs, store design, discount plans and many others.

Evaluation of item sets: Once you have found the frequent itemsets of a dataset, you need to choose a subset of them as your recommendations. Commonly used metrics for measuring significance and interest for selecting rules for recommendations are:

1. **Confidence** (denoted as $\text{conf}(A \to B)$): *Confidence* is defined as the probability of occurrence of B in the basket if the basket already contains A:

$$\text{conf}(A \to B) = \Pr(B|A),$$

   where $\Pr(B|A)$ is the conditional probability of finding item set B given that item set A is present.

2. **Lift** (denoted as $\text{lift}(A \to B)$): *Lift* measures how much more "A and B occur together" than "what would be expected if A and B were statistically independent":

$$\text{lift}(A \to B) = \frac{\text{conf}(A \to B)}{S(B)},$$

   where $S(B) = \frac{\text{Support}(B)}{N}$ and N = total number of transactions (baskets).

3. **Conviction** (denoted as $\text{conv}(A \to B)$): *Conviction* compares the "probability that A appears without B if they were independent" with the "actual frequency of the appearance of A without B":

$$\text{conv}(A \to B) = \frac{1 - S(B)}{1 - \text{conf}(A \to B)}.$$

(a) [3pts]

A drawback of using *confidence* is that it ignores $\Pr(B)$. Why is this a drawback? Explain why *lift* and *conviction* do not suffer from this drawback?

(b) [3pts]

A measure is *symmetrical* if $\text{measure}(A \to B) = \text{measure}(B \to A)$. Are all the measures presented here symmetrical? Explain.

(c) [4pts]

A measure is *desirable* if its value is maximal for rules that hold 100% of the time (such rules are called *perfect implications*). This makes it easy to identify the best rules. Which of the above measures have this property? Explain why.

Product Recommendations: The action or practice of selling additional products or services to existing customers is called *cross-selling*. Giving product recommendation is one of the examples of cross-selling that are frequently used by online retailers. One simple method to give product recommendations is to recommend products that are frequently browsed together by the customers.

Suppose we want to recommend new products to the customer based on the products they have already browsed on the online website. Write a program using the *A-priori* algorithm to find products which are frequently browsed together. Fix the support to $s = 100$ (*i.e.* product pairs need to occur together at least 100 times to be considered frequent) and find itemsets of size 2 and 3.

Use the online browsing behavior dataset at: `http://snap.stanford.edu/class/cs246-data/browsing.txt`. Each line represents a browsing session of a customer. On each line, each string of 8 characters represents the id of an item browsed during that session. The items are separated by spaces.

Note: for parts (d) and (e), the writeup will require a specific rule ordering but the program need not sort the output.

(d) [10pts]

Identify pairs of items (X, Y) such that the support of $\{X, Y\}$ is at least 100. For all such pairs, compute the *confidence* scores of the corresponding association rules: $X \Rightarrow Y$, $Y \Rightarrow X$. Sort the rules in decreasing order of *confidence* scores and list the top 5 rules in the writeup. Break ties, if any, by lexicographically increasing order on the left hand side of the rule.

(e) [10pts]

Identify item triples (X, Y, Z) such that the support of $\{X, Y, Z\}$ is at least 100. For all such triples, compute the *confidence* scores of the corresponding association rules: $(X, Y) \Rightarrow Z$, $(X, Z) \Rightarrow Y$, $(Y, Z) \Rightarrow X$. Sort the rules in decreasing order of *confidence* scores and list the top 5 rules in the writeup. Order the left-hand-side pair lexicographically and break ties, if any, by lexicographical order of the first then the second item in the pair.

What to submit

Upload all the code on snap and include the following in your writeup:

(i) Explanation for 2(a).

(ii) Explanation for 2(b).

(iii) Explanation for 2(c).

(iv) Top 5 rules with confidence scores [2(d)].

(v) Top 5 rules with confidence scores [2(e)].

3 Locality-Sensitive Hashing (15 pts) [Hima/Caroline/You]

An alternative definition for locality-sensitive hashing schemes is:

Definition A locality-sensitive hashing scheme is a set $\mathcal{F}$ of hash functions that operate on a set of objects, such that for two objects x, y,

$$\Pr_{h\in\mathcal{F}}[h(x) = h(y)] = \text{sim}(x, y),$$

where sim is a similarity function that maps a pair of objects (x, y) to a single real number in $[0, 1]$.[1]

In class, we discussed the locality-sensitive hashing schemes for some standard similarity functions like Jaccard similarity and cosine similarity. However, not all similarity functions have a locality-sensitive hashing scheme of this form. In this question, we prove a necessary condition for the similarity function and use it on some simple functions to show that they have no locality-sensitive hashing scheme of this form.

(a) [5pts] Necessary condition

For a similarity function sim to have a locality-sensitive hashing scheme of the form given above, prove that the function $d(x, y) = 1 - \text{sim}(x, y)$ has to satisfy the triangle inequality. (Note: Triangle inequality is $d(x, y) + d(y, z) \geq d(x, z)$, for all x, y, z.)

(b) [5pts]

By means of a counterexample, show that there is no locality-sensitive hashing scheme for the Overlap similarity function:

$$\text{sim}_{\text{Over}}(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)},$$

where A, B are two sets.

[1] Side note: This is a more restrictive definition than that in Section 3.6.1 of the MMDS book: in terms of Fig. 3.9 in the book, our definition here would correspond to the linear function $p = 1 - d$, with $p_1 = p_2$ and $d_1 = d_2$. That is, if the triangle inequality does not hold, you may still obtain an LSH scheme in the sense of the book, but not in the stricter sense we consider in this problem.

(c) [5pts]

When minhashing, one might expect that we could estimate the Jaccard similarity without using all possible permutations of rows. For example, we could only allow cyclic permutations i.e., start at a randomly chosen row r, which becomes the first in the order, followed by rows $r+1$, $r+2$, and so on, down to the last row, and then continuing with the first row, second row, and so on, down to row $r-1$. There are only n such permutations if there are n rows. However, these permutations are not sufficient to estimate the Jaccard similarity correctly. Give an example of two columns such that the probability (over cyclic permutations only) that their minhash values agree is not the same as their Jaccard similarity. In your answer, please provide (a) an example of a matrix with two columns (let the two columns correspond to sets denoted by $S1$ and $S2$) (b) the Jaccard similarity of $S1$ and $S2$ (c) the probability that a random cyclic permutation yields the same minhash value for both $S1$ and $S2$.

What to submit

Include the following in your writeup:

(i) Proof for 3(a)

(ii) Counterexample for 3(b)

(iii) Example for 3(c)

4 LSH for Approximate Near Neighbor Search (30 pts) [Tim/Jacky/Leon]

In this problem, we study the application of LSH to the problem of finding approximate near neighbors.

Assume we have a dataset $\mathcal{A}$ of n points in a metric space with distance metric $d(\cdot,\cdot)$. Let c be a constant greater than 1. Then, the (c,λ)-Approximate Near Neighbor (ANN) problem is defined as follows: Given a query point z, assuming that there is a point x in the dataset with $d(x,z) \le \lambda$, return a point x' from the dataset with $d(x',z) \le c\lambda$ (this point is called a (c,λ)-ANN). The parameter c therefore represents the maximum approximation factor allowed in the problem.

Let us consider a LSH family $\mathcal{H}$ of hash functions that is $(\lambda, c\lambda, p_1, p_2)$-sensitive for the distance measure $d(\cdot,\cdot)$. Let[2] $\mathcal{G} = \mathcal{H}^k = \{g = (h_1, \dots, h_k) | h_i \in \mathcal{H},\ \forall\ 1 \le i \le k\}$, where $k = \log_{1/p_2}(n)$.

[2] The equality $\mathcal{G} = \mathcal{H}^k$ is saying that every function of $\mathcal{G}$ is an AND-construction of k functions of $\mathcal{H}$, so $g(x) = g(y)$ only if $h_i(x) = h_i(y)$ for every h_i underlying g.

Let us consider the following procedure:

1. Select $L = n^\rho$ random members $g_1, \dots, g_L$ of $\mathcal{G}$, where $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$.
2. Hash all the data points as well as the query point using all g_i $(1 \leq i \leq L)$.
3. Retrieve at most[3] $3L$ data points (chosen uniformly at random) from the set of L buckets to which the query point hashes.
4. Among the points selected in phase 3, report the one that is the closest to the query point as a (c, λ)-ANN.

The goal of the first part of this problem is to show that this procedure leads to a correct answer with constant probability.

(a) [5 pts]

Let $W_j = \{x \in \mathcal{A} | g_j(x) = g_j(z)\}$ $(1 \leq j \leq L)$ be the set of data points x mapping to the same value as the query point z by the hash function g_j. Define $T = \{x \in \mathcal{A} | d(x, z) > c\lambda\}$. Prove:

$$\Pr\left[\sum_{j=1}^{L} |T \cap W_j| \geqslant 3L\right] \leqslant \frac{1}{3}.$$

(Hint: Markov's Inequality.)

(b) [5 pts]

Let $x^* \in \mathcal{A}$ be a point such that $d(x^*, z) \leq \lambda$. Prove:

$$\Pr\left[\forall\ 1 \leq j \leq L,\ g_j(x^*) \neq g_j(z)\right] < \frac{1}{\mathrm{e}}.$$

(c) [5 pts]

Conclude that with probability greater than some fixed constant the reported point is an actual (c, λ)-ANN.

[3] If there are fewer than $3L$ data points hashing to the same buckets as the query point, just take all of them.

(d) [15 pts]

A dataset of images,[4] `patches.mat`, is provided in: http://snap.stanford.edu/class/cs246-data/lsh.zip. For this problem, if you don't have matlab on your computer, you may want to use matlab on corn. To do so execute

```
ssh -X <SUNET ID>@corn.stanford.edu
(Your stanford email password)
module load matlab
matlab
```

Each column in this dataset is a 20×20 image patch represented as a 400-dimensional vector. We will use the L_1 distance metric on $\mathbb{R}^{400}$ to define similarity of images. We would like to compare the performance of LSH-based approximate near neighbor search with that of linear search.[5] You should use the code provided with the dataset for this task. The included `ReadMe.txt` file explains how to use the provided code. In particular, you will need to use the functions `lsh` and `lshlookup`. The parameters $L = 10, k = 24$ work for this exercise, but feel free to use other parameter values as long as you explain the reason behind your parameter choice.

- For each of the image patches in columns $100, 200, 300, \ldots, 1000$, find the top 3 near neighbors[6] (excluding the original patch itself) using both LSH and linear search. What is the average search time for LSH? What about for linear search?
- Assuming $\{z_j | \ 1 \leq j \leq 10\}$ to be the set of image patches considered (*i.e.*, z_j is the image patch in column $100j$), $\{x_{ij}\}_{i=1}^{3}$ to be the approximate near neighbors of z_j found using LSH, and $\{x^*_{ij}\}_{i=1}^{3}$ to be the (true) top 3 near neighbors of z_j found using linear search, compute the following error measure:

$$error = \frac{1}{10} \sum_{j=1}^{10} \frac{\sum_{i=1}^{3} d(x_{ij}, z_j)}{\sum_{i=1}^{3} d(x^*_{ij}, z_j)}$$

  Plot the error value as a function of L (for $L = 10, 12, 14, \ldots, 20$, with $k = 24$). Similarly, plot the error value as a function of k (for $k = 16, 18, 20, 22, 24$ with $L = 10$). Briefly comment on the two plots (one sentence per plot would be sufficient).

- Finally, plot the top 10 near neighbors found[7] using the two methods (using the default $L = 10, k = 24$ or your alternative choice of parameter values for LSH) for the image

[4] Dataset and code adopted from Brown University's Greg Shakhnarovich

[5] By linear search we mean comparing the query point z directly with every database point x.

[6] Sometimes, the function `nnlsh` may return less than 3 nearest neighbors. You can use a `while` loop to check that `lshlookup` returns enough results, or you can manually run the program multiple times until it returns the correct number of neighbors.

[7] Same remark, you may sometimes have less that 10 nearest neighbors in your results; you can use the same hacks to bypass this problem.

patch in column 100, together with the image patch itself. You may find the functions `reshape()` and `mat2gray()` useful to convert the matrices to images; you can also use the functions `imshow()` and `subplot()` to display the images.
How do they compare visually?

What to submit

(i) Include the proof for 4(a) in your writeup.

(ii) Include the proof for 4(b) in your writeup.

(iii) Include the reasoning for why the reported point is an actual (c, λ)-ANN in your writeup [4(c)].

(iv) Include the following in your writeup for 4(d):

- Average search time for LSH and linear search.
- Plots for error value vs. L and error value vs. K, and brief comments for each plot
- Plot of 10 nearest neighbors found by the two methods (also include the original image) and brief visual comparison

(v) Upload the matlab code for 4(d) on snap.