CS246: Mining Massive Data Sets **Winter 2013**

Problem Set 1

Due 9:30am January 24, 2013

Only one late period is allowed for this homework (01/29).

General Instructions

These questions require thought, but do not require long answers. Please be as concise as possible. Please fill the cover sheet (`http://cs246.stanford.edu/cover.pdf`) and submit it as a front page with your answers. Include the names of your collaborators (see the course website for the collaboration or honor code policy).

Regular (non-SCPD) students should submit hard copies of the answers and upload the source code (see course website FAQ for further instructions).

SCPD students should submit their answers through SCPD and upload the code via the submission website. The submission must include the usual SCPD routing form available at `http://scpd.stanford.edu/generalInformation/pdf/SCPD_HomeworkRouteForm.pdf`.

Questions

1 MapReduce (20 points) [Le/Dingyi]

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 friends with B then B is also friends with A.

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 most 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 in decreasing number of mutual friends. Even if a user has less than 10 second-degree friends, output all of them in decreasing order of the number of mutual friends.

What to submit: (1) Submit the source code via the electronic submission website. **(2)** As a part of your homework writeup submit the recommendations for the users with following user IDs: 924, 8941, 8942, 9019, 9020, 9021, 9022, 9990, 9992, 9993.

2 Association Rules (30 pts) [Justin / Sumit]

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:

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

   where $P(B|A)$ 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.

3. **Conviction** (denoted as $\text{conv}(A \to B)$): *Conviction* compares the "probability that A appears without B if they were dependent" 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) [2pts]

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

(b) [2pts]

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

(c) [3pts]

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.

(d) [5pts]

List the top 5 rules in decreasing order of *confidence* score for itemsets of size 2. Do it also for itemsets of size 3. (Ties can be broken arbitrarily.)

(e) [5pts]

List the top 5 rules in decreasing order of *lift* score for itemsets of size 2. Do it also for itemsets of size 3. (Ties can be broken arbitrarily.)

(f) [5pts]

List the top 5 rules in decreasing order of *conviction* score for itemsets of size 2. Do it also for itemsets of size 3. (Ties can be broken arbitrarily.)

(g) [2pts]

From all the sets of rules you found in parts (d), (e) and (f), list all the rules (for itemsets of size 2) which hold 100% of the time (maximal for *perfect implications*)? *Hint: Apply concepts from part (c).*

What to submit: A numbered list of rules in the format: [$A \Rightarrow B$; conf, lift, or conv score] for size 2 and [$(A, B) \Rightarrow C$; conf, lift, or conv score] for size 3, with $A \cap B = \emptyset, |A| \geq 1, |B| \geq 1$ and $|A| + |B| = 2$ or 3 depending upon the item set size. Include only one rule if you get symmetric rules (*i.e* include only $A \Rightarrow B$ if you get both $A \Rightarrow B$ and $B \Rightarrow A$ with same scores).

Using A-priori output: Answer the following questions based on the information from the previous questions. Justify your answers.

(h) [2pts]

Which products would you recommend online on the page for product **GRO85051**? State which measure and rule you used for your recommendation.

(i) [2pts]

Which products would you recommend if we know that the customer has browsed the following products: **ELE17451, DAI92600, GRO85051**? Does your recommendation change if a different measure is used?

These kind of recommendations are called personalized recommendations as they are based on purchase and browsing history of a specific customer.

(j) [2pts]

Retailers commonly offer deals at the checkout counter at the store. The counter clerks are only able to suggest one or two recommendations. Which products would you recommend to a customer at the checkout counter if the customer has the following products in the cart: **ELE92920, DAI93865, DAI88079, DAI23334**? Would you make your recommendations based on 2- or 3-itemsets? Explain which would you choose and why.

3 Locality Sensitive Hashing (15 points) [Anshul/Rose]

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,

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

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

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 $d(x, y) = 1 - \mathrm{sim}(x, y)$ has to satisfy the triangle inequality. (Hint: 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:

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

where A, B are two sets.

(c)[5pts]

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

$$\mathrm{sim}_{Dice}(A, B) = \frac{|A \cap B|}{\frac{1}{2}(|A| + |B|)},$$

where A, B are two sets.

4 LSH for Approximate Near Neighbor Search (30 points) [Tongda, Robi]

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 $d(\cdot,\cdot)$. Consider the (c,λ)-Approximate Near Neighbor (ANN) problem: Given a query point z, assuming there is a point x in the dataset with $d(x,z) \leq \lambda$, return a point x' from the dataset with $d(x',z) \leq c\lambda$. Here $c > 1$ is the maximum approximation factor allowed in the problem.

Assume the LSH family $\mathcal{H}$ of hash functions is $(\lambda, c\lambda, p_1, p_2)$-sensitive for the distance measure $d(\cdot,\cdot)$. Let $\mathcal{G} = \mathcal{H}^k = \{g = (h_1,\ldots,h_k) | h_i \in \mathcal{H}\}$, where $k = \log_{1/p_2}(n)$. We take $L = n^\rho$ random members $g_1,\ldots,g_L$ of $\mathcal{G}$, where $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$, and hash all the data points as well as the query point using all g_i's $(1 \leq i \leq L)$. Then, to find an approximate near neighbor, we retrieve at most $3L$ data points from the buckets $g_j(z)$ $(1 \leq j \leq L)$, and report the closest one as a (c,λ)-ANN. We would like to prove that the reported answer is correct, with constant probability.

(a) [5 points]

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:

$$Pr[\sum_{j=1}^{L} |T \bigcap W_j| > 3L] < \frac{1}{3}.$$

(Hint: Markov's Inequality.)

(b) [5 points]

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

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

(c) [5 points]

Conclude that with a constant probability the reported point is an actual (c,λ)-ANN.

(d) [15 points]

A dataset of images[1], `patches.mat`, is provided in:
`http://snap.stanford.edu/class/cs246-data/lsh.zip`

Each column in this dataset is a 20×20 image patch represented as a 400-dimensional vector. We would like to compare the performance of LSH-based approximate near neighbor search with that of linear search. 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`. We will use the L_1 distance measure, and the corresponding LSH with $L = 10, k = 24$. Feel free to use other parameter values, but make sure you explain the reason behind your parameters' choice.

- For each of the image patches in columns $100, 200, 300, \ldots, 1000$, find the top 3 near neighbors[2] (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 $L = 10$).

- Finally, plot the top 10 near neighbors found[3] using the two methods (using the default $L = 10, k = 24$ or your alternative choice of parameter values for LSH) for the image 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?

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

[2] 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.

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