COMP9727 Recommender Systems
Assignment 1 - 22T2

Due: 1st July, 17:00 AEST

Total Mark: 50

Introduction In this assignment, you will be required to manually implement a few recommendation algorithms in Python as well as answer some corresponding questions **individually**. Other than this spec, all the required files can be found in `'a1.zip'`, which you can download directly from the WebCMS3 page. The following are some general requirements for the assignment:

- File link (login required): `https://cs9727.web.cse.unsw.edu.au/22T2/assn1/a1.zip`. If you can not download the file, try to open it in incognito window.
- There are 5 questions that count for 25% of your final mark. They are not equally distributed.
- Each question consists of a **conceptual questions** part and an **implementation** part.
- For **conceptual questions** part, you must submit your answers electronically. You should report your answers to all the questions in a single file `ass1_YourZId.pdf`. You are strongly encouraged to use LaTeX to produce your answers (Word is fine if you prefer). Please note that we do not accept scanned copies.
- For **implementation** part, please note that all the algorithms are required to be implemented in `Python3`.
- As far as third-party packages are concerned, only the following are permitted: `NumPy`, `Pandas` and `Matplotlib`. For versioning, please check it in the submission section.
- For `'a1.zip'`, it contains the following datasets: `q1.txt`, `q2.txt`, `q3.csv`, `shows.txt`, `user-shows.txt`, `q5.csv`. It also contains some starter codes: `q3.py` and `q5.py`, please follow the instructions inside to finish the tasks. For `q1.py`, `q2.py` and `q4.py`, you need to implement them from scratch, including creating the python files yourself.
- Please ensure your code works well on CSE machines. If your code was unable to run on CSE machines, you will receive 0 for the corresponding question.
- Late Penalty: 5% reduction per day of the assessment mark. Late submissions that are more than 5 days late will not be accepted.
- We do not set up standard outputs for the questions. As such, you will not lose marks due to the output format, as long as you include all the required content and the outputs are well-structured and meaningful.

Question 1 - Associate Rules and its Application in Recommendation (10 Marks)

To measure significance and interest in rules for recommendations, there are three commonly used metrics:

- **Confidence** (denoted as $\text{conf}(X \to Y)$): Confidence is defined as the probability of occurrence of Y in the transaction if the transaction already contains X:

$$\text{conf}(X \to Y) = \Pr(Y|X),$$

  where $\Pr(Y|X)$ is the conditional probability.

- **Lift** (denoted as $\text{lift}(X \to Y)$): Lift is the ratio of the confidence of the rule and the expected confidence of the rule if X and Y are statistically independent:

$$\text{lift}(X \to Y) = \frac{\text{conf}(X \to Y)}{\text{support}(Y)};$$

- **Conviction**(denoted as $\text{conv}(X \to Y)$): Conviction can be explained as the expected frequency that X occurs without Y (i.e., the frequency that the rule makes an incorrect prediction):

$$\text{conv}(X \to Y) = \frac{1 - \text{support}(Y)}{1 - \text{conf}(X \to Y)}$$

(a) (2 Marks)

There is a problem by using confidence only to evaluate the rules: confidence ignores the $\Pr(Y)$. Explain why it matters. Moreover, explain why lift and conviction are not affected by that.

(b) (2 Marks)

Consider the following definition of symmetrical:

Definition 1 (Symmetrical) *A measure is symmetrical if measure*$(X \to Y) =$ *measure*$(Y \to X)$*.*

For those mentioned measures: Confidence, Lift and Conviction, prove or disprove that they are symmetrical.

(c) (2 Marks)

A rule is said to be a "perfect" rule if it has a conditional probability of 1 (i.e., $Pr(Y|X) = 1$ for $X \to Y$). A measure is considered *optimal* if it reaches its maximum achievable value for a "perfect" rule. Which of these measures is *optimal*? You should prove or disprove the statement. You can ignore $0/0$ but not other cases.

Now let's move to the implementation part. You are required to create a program named `q1.py`.
Use your program to answer the following questions: (d) and (e).
We will use the provided dataset `q1.txt` to apply the associate rule mining technique to make recommendations. In this dataset, each line represents a session of a user while the user id is not

known. Within each session, there are several item IDs (strings of length 8) that the user viewed during this session, separated by space.
In your `q1.py`, use the *Apriori algorithm* to find the items which are frequently viewed together. Assume the support is 100 and fixed. Find the itemsets of sizes 2 and 3.

(d) (2 Marks)

Given a pair of items (X, Y), assume that the support of $\{X, Y\}$ is at least 100. Compute the confidence scores of the following two association rules for all the pairs meeting that condition:

$$X \to Y, \quad Y \to X$$

Sort the rules in **decreasing order based on confidence scores**. Additionally, if more than one rule has the same confidence score, sort them by **lexicographically increasing order on the left hand side of the rule**. Please report top 10 rules in your report. Note that, your program should output top 20 rules for sanity check.

(e) (2 Marks)

Now consider a tuple of items (X, Y, Z), follow the same instructions in part (d) and report your results. The association rules you need to consider are:

$$(X, Y) \to Z, \ (X, Z) \to Y, \ \text{ and } \ (Y, Z) \to X$$

Question 2 - Latent Factor Method (5 Marks)

The goal of this question is to implement an algorithm - *Stochastic Gradient Descent (SGD)* to build a recommender system. Given a matrix R of ratings which contains m items and n users (i.e., size of R is $m \times n$). R_{iu} represents the rating given by user u to item i. The goal of this problem is to find user matrix P and item matrix Q. We define the error as the following:

$$E = \left(\sum_{(i,u) \in R} (R_{iu} - \hat{R}_{iu})^2 \right) + \lambda \left[\sum_u \|p_u\|_2^2 + \sum_i \|q_i\|_2^2 \right], \text{ where } \hat{R}_{iu} = q_i \cdot p_u^\top \tag{1}$$

In the lecture, we've discussed how SGD is used for updating the gradient. Now, implement the algorithm in a file named `q2.py` by using the provided dataset `q2.txt` with the following requirements.

- You are **not allowed** to store the matrix R in memory. You have to read each R_{iu} once a time from disk and apply your update rules in each iteration.
- Your program should read the whole file within each iteration.
- There is no code template provided in this question, you need to implement it from scratch.

Now, let's fix $k = 20, \lambda = 0.1$ and the number of iterations $= 40$. You are tasked with finding an optimal value for the learning rate η. With an ideal η, you should observe the following: $E < 65,000$ after 40 iterations with both q_i and p_u stop changing.

Find the value of η that you believe is ideal. Based on the η you find, please plot the corresponding value of the objective function E (defined in Eq. 1) changing over the number of iterations (i.e., value of E as y-axis and iteration as x-axis).

Question 3 - Collaborative Filtering with Naive Bayes Classifier (5 Marks)

Collaborative filtering (CF) can be viewed as a generalization of classification tasks when there are a few distinct unordered **discrete** ratings. In this question, you will be required to work with collaborative filtering based on Naive Bayes classifier (NBC). Built upon the Bayes' theorem, NBC is a simple yet effective supervised classification algorithm with conditional independence assumption. Concretely, consider a K-class classification problem where an instance $\mathbf{x} = \{x_i\}_{i=1:d}$ is represented by a d-dimensional feature vector. The probability that it belongs to each of K possible outcomes is given by

$$\underbrace{\Pr\left(C_k \mid \mathbf{x}\right)}_{\text{posterior}} = \frac{\overbrace{\Pr\left(\mathbf{x} \mid C_k\right)}^{\text{likelihood}} \overbrace{\Pr\left(C_k\right)}^{\text{prior}}}{\underbrace{\Pr\left(\mathbf{x}\right)}_{\text{evidence}}}, \quad k = 1 \dots K \tag{2}$$

Since the evidence $\Pr(\mathbf{x})$ does not depend on C, we only care about the numerator for different C_k in practice. Under a "naive" assumption that features are mutually independent conditioned on the class C_k - $\Pr(x_i \mid x_{i+1}, \dots, x_d, C_k) = \Pr(x_i \mid C_k)$, Eq. 2 can be re-written as

$$\begin{aligned}
\Pr\left(C_k \mid \mathbf{x}\right) &\propto \Pr\left(\mathbf{x} \mid C_k\right) \Pr\left(C_k\right) \\
&\propto \Pr\left(\mathbf{x}, C_k\right) \\
&\propto \Pr\left(x_1 \mid x_2, \dots, x_d, C_k\right) \Pr\left(x_2 \mid x_3, \dots, x_d, C_k\right) \cdots \Pr\left(x_d \mid C_k\right) \Pr\left(C_k\right) \\
&\propto \Pr\left(x_1 \mid C_k\right) \Pr\left(x_2 \mid C_k\right) \cdots \Pr\left(x_n \mid C_k\right) \Pr\left(C_k\right) \\
&\propto \prod_{i=1}^{n} \Pr\left(x_i \mid C_k\right) \Pr\left(C_k\right)
\end{aligned} \tag{3}$$

Now let's take a look at a concrete recommendation scenario. Assume there is a rating matrix $R \in r^{m \times n}$ associated with a user set $\mathcal{U} = \{u_l\}_{l=1:m}$ and an item set $\mathcal{I} = \{i_l\}_{l=1:n}$ consisting of m users and n items, respectively. The (u, i)-th entry of R represents the specified rating of the u-th user to the i-th item, denoted by $r_{u,i} = \{0, 1\}$, i.e., rating is a random binary variable. In this case, estimating absent ratings is cast to a classification task. As usual, one can perform either *user-based* CF or *item-based* CF, with the differences shown by how prior and likelihood are computed. You may find the following steps to perform CF with NBC.

1. Prior and Likelihood Computation. In *user-based* CF, prior and likelihood are computed according to the items that each user has rated. Specifically, let $\Pr(r_{\cdot,i} = k)$ be the prior probability of item i being rated as k, defined as

$$\Pr\left(r_{\cdot,i} = k\right) = \frac{\mathbf{card}\left\{u \in \mathcal{U} \mid r_{u,i} = k\right\} + \alpha}{\mathbf{card}\left\{u \in \mathcal{U} \mid r_{u,i} \neq \text{null}\right\} + \alpha \cdot \mathbf{card}\left\{R\right\}} \tag{4}$$

where $\mathbf{card}\{\cdot\}$ counts the number of elements in the set, $\mathbf{card}\{R\}$ represents all plausible ratings (i.e., not empty and valid value) in the rating matrix. null denotes the absence of rating, and α refers to Laplacian smoothing parameter to handle over-fitting. Let $\Pr(r_{\cdot,j} = k' \mid r_{\cdot,i} = k)$ be the probability of item j being rated as k' **conditioning** on $r_{\cdot,i} = k$, then

$$\Pr\left(r_{\cdot,j} = k' \mid r_{\cdot,i} = k\right) = \frac{\mathbf{card}\left\{u \in \mathcal{U} \mid r_{u,j} = k' \wedge r_{u,i} = k\right\} + \alpha}{\mathbf{card}\left\{u \in \mathcal{U} \mid r_{u,j} \neq \text{null} \wedge r_{u,i} = k\right\} + \alpha \cdot \mathbf{card}\left\{R\right\}} \tag{5}$$

As for *item-based* CF, prior and likelihood are computed according to the ratings that each item has received. Analogously, define $\Pr(r_{u,\cdot} = k)$ as the prior probability that user u rates an

arbitrary item with value k

$$\Pr\left(r_{u,\cdot}=k\right)=\frac{\textbf{card}\left\{i\in\mathcal{I}\mid r_{u,i}=k\right\}+\alpha}{\textbf{card}\left\{i\in\mathcal{I}\mid r_{u,i}\neq\text{null}\right\}+\alpha\cdot\textbf{card}\left\{R\right\}} \tag{6}$$

Meanwhile, let $\Pr\left(r_{v,\cdot}=k'\mid r_{u,\cdot}=k\right)$ be the probability that user v rate k' to an item **conditioning** on $r_{u,\cdot}=k$,

$$\Pr\left(r_{v,\cdot}=k'\mid r_{u,\cdot}=k\right)=\frac{\textbf{card}\left\{i\in\mathcal{I}\mid r_{v,i}=k'\wedge r_{u,i}=k\right\}+\alpha}{\textbf{card}\left\{i\in\mathcal{I}\mid r_{v,i}\neq\text{null}\wedge r_{u,i}=y\right\}+\alpha\cdot\textbf{card}\left\{R\right\}} \tag{7}$$

2. Rating Estimation. Using prior and factors of the likelihood, we are able to derive the classification score $\Pr\left(r_{u,i}=k\right)$ of user u to the item i of being a specific rating k.

$$\Pr\left(r_{u,i}=k\right)\propto\begin{cases}\Pr\left(r_{\cdot,i}=k\right)\prod_{j\in\mathcal{I}_u}\Pr\left(r_{\cdot,j}=r_{u,j}\mid r_{\cdot,i}=k\right), & \text{if user-based}\\ \Pr\left(r_{u,\cdot}=k\right)\prod_{v\in\mathcal{U}_i}\Pr\left(r_{v,\cdot}=r_{v,i}\mid r_{u,\cdot}=k\right), & \text{if item-based}\end{cases} \tag{8}$$

where $\mathcal{I}_u=\{i\in\mathcal{I}\mid r_{u,i}\neq\text{null}\}$ collects the items that user u has rated. $\mathcal{U}_i=\{u\in\mathcal{U}\mid r_{u,i}\neq\text{null}\}$ is the set of users who have rated item i.

Finally, the predicted rating is produced by the one that maximizes the classification score,

$$\hat{r}_{u,i}=\arg\max_k\Pr\left(r_{u,i}=k\mid R\right) \tag{9}$$

(a) (2 Marks)

Assume, an interaction matrix R described by 5 users and 5 items where each entry is a binary value denoting the implicit feedback. Please derive a *user-based* CF for $\hat{R}_{2,1}$ and an *item-based* CF for $\hat{R}_{4,5}$, by following the strategies of NBC above.

$$R=\begin{pmatrix}0&?&1&0&1\\?&1&0&?&1\\0&0&1&1&1\\1&0&1&?&?\\1&1&0&0&1\end{pmatrix}$$

For each entry, you are supposed to show the classification score of each possible rating, as well as the predictions (w.r.t Eq. 8 and Eq. 9). You can directly use the multiplication result as the score (i.e., replace $\propto$ with $=$ in Eq. 8). The Laplacian smoothing coefficient here $\alpha=0.1$.

Your answer should show how you derive the results.

(b) (3 Marks)

Upon knowing how *user-based* CF and *item-based* CF work, you can also combine them into a hybrid one. It goes without saying that considering them both would increase the use of evidence from R. As such, the classification score of $r_{u,i}=k$ becomes the weighed product of both sides

$$\begin{aligned}\Pr\left(r_{u,i}=y\right)\propto&\left(\Pr\left(r_{\cdot,i}=k\right)\prod_{j\in\mathcal{I}_u}\Pr\left(r_{\cdot,j}=r_{u,j}\mid r_{\cdot,i}=k\right)\right)^{\frac{1}{1+\textbf{card}\{\mathcal{U}_i\}}}\\&\cdot\left(\Pr\left(r_{u,\cdot}=k\right)\prod_{v\in\mathcal{U}_i}\Pr\left(r_{v,\cdot}=r_{v,i}\mid r_{u,\cdot}=k\right)\right)^{\frac{1}{1+\textbf{card}\{\mathcal{I}_u\}}}\end{aligned} \tag{10}$$

where $\mathcal{U}_i$ and $\mathcal{I}_u$ represents the set of users who have rated item i, and the set of items that have been rated by user u, respectively.

Now given another rating matrix R to be read from a file `q3.csv`, you are tasked with implementing a hybrid CF that enables rating predictions of a specified missing entry. Given the fact that, the prediction performed by a hybrid CF (w.r.t Eq. 10) is simply a weighted production of *user-based* CF and *item-based* CF, you should implement them as well in your program.

For each specified query to be predicted, your program should be able to estimate the classification score for each possible rating value, and predict the rating value accordingly.

Note:

- **You are provided with an example dataset `q3.csv` and the starter code `q3.py` where the helper functions are placed (DO NOT ALTER THEM).**
- **Complete the program ONLY within the indicated area and focus on the core functionalities. You should follow instructions inside `q3.py`.**
- **Do NOT hardcode the results since the testing data may be different from the provided example dataset.**

Question 4 - User-item bipartite graph in Recommendation (15 Marks)

Let's consider a common data structure - graph. We define a user-item bipartite graph proposed by Li and Chen (2013). The user-item bipartite graph in this question is split into two separate and disjoint sets U and I representing users and items, respectively. In a bipartite user-item graph, an edge indicates that user u likes item i or that user u provides positive feedback to item i. We also have the rating matrix R where each row represents a user and each column corresponds to an item. If user u likes item i then $R_{u,i} = 1$, otherwise $R_{u,i} = 0$. We assume that there are m users and n items.
Define an m-by-m diagonal matrix P, where each diagonal element represents the corresponding degree of the user node. Similarly, each diagonal element in an n-by-n matrix Q represents the degree of each item node. See the following figure for an example.

(a) (2 Marks)

Now we define a user related matrix $T = RR^\top$. Describe what T_{ii} and T_{ij} $(i \neq j)$ represent respectively in the above mentioned bipartite graph.

(b) (3 Marks)

Now we define an item similarity matrix S_I, with a size of $n \times n$. Each element represents the pairwise cosine similarity between items. For example, S_I^{ij} represents the element in row i and column j, which is the cosine similarity between item i and item j.

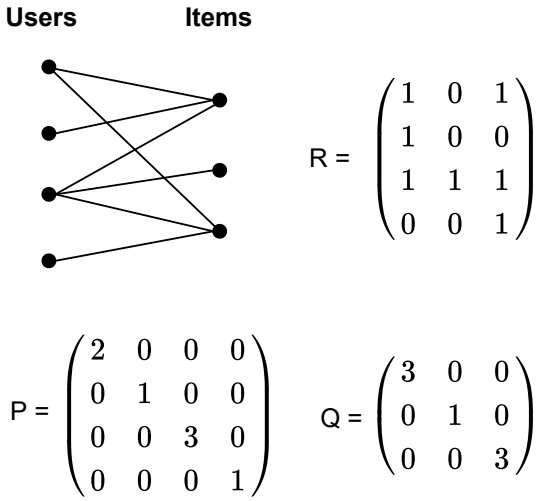

Figure 1: Example of bipartite graph

Show that $S_I = Q^{-1/2} R^\top R Q^{-1/2}$.
Hint: row i and column j in S_I correspond to column i and column j of the matrix R, respectively.

Now consider the user similarity matrix S_U, with a size of $m \times m$. Please find the representation of S_U by using R, P and Q.

Your answer should show how you derive the expressions.

(c) (4 Marks)

Let's define the final recommendation matrix M, with a size of $m \times n$, such that $M(i, j) = r_{i,j}$. Find M for **both** item-based and user-based collaborative filtering approaches (assume the cosine similarity is used) by using R, P and Q. Your final answer should describe operations at the matrix level, not specific terms of matrices.

Different from the lecture, we will use a simplified version to represent rating prediction for user u and item s, $r_{u,s}$:

$$r_{u,s} = \sum_{x \in users} R_{x,s} \cdot sim(x, u) \text{ and } r_{u,s} = \sum_{x \in items} R_{u,x} \cdot sim(x, s)$$

Your answer should show how you derive the expressions.

(d) (6 Marks)

Now, let's apply those methods into the a real application. Given a dataset that contains the information about TV shows. More precisely, for $9,985$ users and 563 popular TV shows, we know if a given user watched a given show over a 3 month period. You are provided with two files: `user-shows.txt` and `shows.txt`:

- `user-shows.txt`: This is the ratings matrix R, where each row corresponds to a user and each column corresponds to a TV show. $R_{ij} = 1$ if user i watched the show j over a period of three months. The columns are separated by a space.

– `shows.txt`: This is a file containing the titles of the TV shows, in the same order as the columns of R.

We will compare the user-based and item-based collaborative filtering recommendations for the 500-th user of the dataset. Let's call him Bob. In order to do so, we have erased the first 100 entries of Bob's row in the matrix, and replaced them by 0s. This means that we don't know which of the first 100 shows Bob has watched. Based on Bob's behaviour on the other shows, we will give Bob recommendations on the first 100 shows. We will then see if our recommendations match what Bob had in fact watched.

Here are the tasks:

(1) Compute P and Q. (2 Marks)

(2) Based on your results about M in part (c), compute M for the user-based collaborative filtering. Let S denote the set of the first 100 shows (the first 100 columns of the matrix). What are the **five** TV shows in S that have the highest similarity scores for Bob? Choose the show with the smaller index if similarity scores are tied between two shows. Do not write the index of the TV shows. Write their names using the file `shows.txt`. (2 Marks)

(3) Compute the matrix M for the item-based collaborative filtering. From all the TV shows in S, which are the **five** that have the highest similarity scores for Bob? In case of ties between two shows, choose the one with smaller index. Again, report the names of the shows and their similarity scores. (2 Marks)

Note: Write a program, stored in a file named `q4.py`. You program `q4.py` should produce the following four outputs in order:

– **Matrix P**

– **Matrix Q**

– **Top 5 items for Bob that have the highest similarity scores by using user-based collaborative filtering**

– **Top 5 items for Bob that have the highest similarity scores by using item-based collaborative filtering.**

Matrix can be printed as a `numpy.ndarray`. If your code cannot produce the correct output (i.e., same as your report), you will get 0 for this part.

Question 5 - Locality Sensitive Hashing for ANN Search (15 Marks)

In Lecture 4, we've discussed LSH and ANN. Now, we would like to further study how to use LSH to conduct the ANN search.

Assume we have a dataset $\mathcal{A}$ consisting of n points. Let's define a distance metric $d(\cdot, \cdot)$, and a constant $c : c > 1$. Now define a (c, λ)-ANN problem as the following: Given an arbitrary query data point z, assuming there exists a point x such that $d(x, z) \leq \lambda$, return a point y from the dataset such that $d(y, z) \leq c\lambda$. The point y is known as (c, λ)-ANN. Hence, c here presents the maximum allowable approximation factor.

Now let us consider a LSH family $\mathcal{H}$ that is $(\lambda, c\lambda, p_1, p_2)$-sensitive with the distance measure $d(\cdot, \cdot)$. Let $\mathcal{G} = \mathcal{H}^k = \{g = (h_1, h_2, \cdots, h_k) \mid h_i \in \mathcal{H}, \forall 1 \leq i \leq k\}$, where $k = \log_{1/p_2}(n)$. Now we have the following procedure:

- Assume we have $N = n^p$ random numbers from $\mathcal{G} : g_1, g_2, \cdots, g_N$, with $p = \frac{\log(1/p_1)}{\log(1/p_2)}$.
- Hash all the data as well as the query data by using $g_i (i \in [1, N])$.
- Retrieve $3N$ (at most) data points from the set of N buckets to which the query point hashes. (Note: It may occur that we cannot retrieve $3N$ data points in some cases. In that case, you should fetch all the data points.)
- Select the data points closest to the query point from your $3N$ data points. Return it as the (c, λ)-ANN.

Before we proceed with the implementation, we would like to prove that the above procedure can produce a correct answer with a certain probability.

(a) (2 Marks)

Let's define two sets: $B_j = \{x \in \mathcal{A} \mid g_j(x) = g_j(z)\}$ where $j \in [1, N]$, and $C = \{x \in \mathcal{A} \mid d(x, z) > c\lambda\}$. Prove that:

$$Pr\left[\sum_{j=1}^{N} |B_j \cap C| \geq 3N\right] \leq \frac{1}{3}$$

Hint: Markov's inequality

(b) (3 Marks)

Let $y \in \mathcal{A}$ be the point that $d(y, z) \leq \lambda$. Prove that:

$$Pr\Big[g_j(y) \neq g_j(z)\Big] < \frac{1}{e}, \quad \text{for } \forall\, j,\ 1 \leq j \leq N$$

(c) (4 Marks)

From your results in the previous part, prove that the report point with probability greater than some fixed constant is an actual (c, λ)-ANN.

Hint: Try to prove it by finding two ways that the point is not a (c, λ)-ANN.

(d) (6 Marks)

We now move on to implementation. Consider a dataset `q5.csv`, where each row is a 20×20 image, represented as a flattened 400-dimensional vector. For the similarity, we will use the Manhattan Distance. We are interested in the performance between LSH and linear search (i.e., we compare the query data z directly with every data point x in the dataset). The default setting is that $N = 10, k = 24$.

By using the starter code `q5.py`, you are required to conduct the following tasks:

(1) For each image in row: `range(100,1001,100)`, find the top 3 nearest neighbours (excluding the origin data point) by using LSH and linear search, respectively. For both LSH and linear search, what is the average search time? Record them in your report. (2 Marks)

(2) Assume we have $\{z_j \mid j \in [1, 10]\}$ to be the indices that we would like to consider for retrieving images (i.e., same as the previous part, in this part we only consider the 100j-th images). Now we have $\{x_{ij}\}_{i=1}^{3}$ be the top-3 ANN of z_j found by LSH,

and $\{x_{ij}^*\}_{i=1}^3$ be the true top-3 ANN of z_j found by linear search. Now we define the following error function:

$$\mathcal{L} = \frac{1}{10}\sum_{j=1}^{10}\frac{\sum_{i=1}^{3} d(x_{ij}, z_j)}{\sum_{i=1}^{3} d(x_{ij}^*, z_j)}.$$

Let's consider the error value as a function of N. In this case, we are interested in seeing how the error values will be affected by N. Let's assume that $N =$`range(10,21,2)` and $k = 24$. Please plot the corresponding function. (i.e., error value as y-axis and N as x-axis)

In addition, we also want to see how error values will be affected by k. Let's assume that $k =$`range(16,25,2)` and $N = 10$. Please plot error value as a function of k.

Your findings should be described in one sentence each. (2 Marks)

(3) For the image in row 100, please plot the top 10 nearest neighbours found using two methods – LSH and linear search. You can use the default setting for LSH ($N = 10, k = 24$), or you can determine it by yourself. Note: Please plot the image in row 100 itself as well. (2 Marks)

Note: There are several TODO tasks in the starter code that you need to complete. You will get 0 for the entire part (d) if your code was unable to produce the results in your report.

Submission

You should submit by typing the following command in CSE machine

```
give cs9727 ass1 ass1_YourZId.pdf q1.py q2.py q3.py q4.py q5.py
```

*Keep in mind that your code must be compatible with the **default** environment on any CSE machine, including AND limited to*

```
Python 3.7.3    Numpy 1.12.5   pandas 0.23.3   Matplotlib 3.0.2
```

Plagiarism

Group submissions will not be allowed. Your program must be entirely your own work. Plagiarism detection software will be used to compare all submissions pairwise (including submissions for similar assignments in previous years, if applicable) and serious penalties will be applied, particularly in the case of repeat offenses. Do not copy ideas or code from others. Do not use a publicly accessible repository or allow anyone to see your code.

Marking Criteria

Question 1

(a) Correctly explain why confidence ignore the $Pr(Y)$ matters (1 Mark). Explain why lift and conviction not affected (0.5 Mark each).

(b) Get one correct for 1 Mark, two for 1.5 Marks and three for 2 Marks.

(c) Get one correct for 1 Mark, two for 1.5 Marks and three for 2 Marks.

(d) Correct results for 1 Mark, correct order for 1 Mark.

(e) Correct results for 1 Mark, correct order for 1 Mark.

Question 2

Correct Implementation (3 Marks), correct plot (1 Mark), correct η (1 Mark). If you store the matrix in memory, 0 will be awarded for implementation part.

Question 3

(a) Correct classification scores (round up to 2 decimal places) and predicted ratings (2 Marks)

(b) Reasonable implementation (Not hard-code), including correct classification scores (round up to 2 decimal places) and predicted rating for *user-based* CF (1 Mark), *item-based* CF (1 Mark), and hybrid CF (1 Mark).

Question 4

(a) Correctly explain meaning of T_{ii} and T_{ij} (1 Mark each).

(b) Correctly show the result of S_I for 1.5 Marks and the representation fo S_U for 1.5 Marks.

(c) M for item-based CF and user-based CF (2 Marks each).

(d) P and Q (1 Mark each); top 5 items for user-based CF and item-based CF (2 Marks each).

Question 5

(a) Correct proof (2 Marks).

(b) Correct proof (3 Marks).

(c) Find two ways that point is not (c, λ)-ANN (1.5 Marks each), and a conclusion for each (1 Mark).

(d) average search time for LSH and linear search (1 Mark each); Two plots and their descriptions (0.5 Mark each); Plot of the top 10 NN found by LSH and linear search (1 Mark each).

References

Xin Li and Hsinchun Chen. 2013. Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems* 54, 2 (2013), 880–890.