18734 Homework 4

Due: 12 noon Eastern, 9 am Pacific, Nov 14

Problem 1: l-diversity and t-closeness (25 marks)

We discussed an attack on k-anonymity in class, which led to the development of l-diversity. However, l-diversity also suffers from an attack, which you will explore in the first part of this question. t-closeness is a further refinement of l-diversity.

1. Read the intersection attack and the definition of l-diversity in this paper `http://www.cse.psu.edu/~kasivisw/kdd.pdf`. First, explain what l-diversity means. It may help to choose a small example table and choose values of l and s from the definition of l-diversity to check the requirements of being l-diverse. Next, explain why the intersection attack works against l-diversity by showing an example. (The example in the paper is not using an l-diverse database, but, it is possible to construct one).

2. Next, look at the paper on t-closeness. (`http://www.cs.purdue.edu/homes/ninghui/papers/t_closeness_icde07.pdf`) It talks about shortcomings of l-diversity. Explain the skewness and similarity attacks that the paper describes, by showing examples different from the one in the paper.

Problem 2: Laplace mechanism (25 marks)

In this question, we will take a closer look at a mechanism for achieving ϵ-differential privacy for query functions $count_P$ (which returns the number of entries in a database with property P) and $max - salary - at - Berkeley$ (which returns the maximum salary of faculty and staff at UC Berkeley). This mechanism, denoted κ_f for query function f, computes $f(X)$ and adds noise with a Laplace distribution with variance σ^2 (note variance is the whole square of standard deviation) that depends on the sensitivity of the function and the differential privacy parameter ϵ. State precisely any additional assumptions you make to answer the questions below.

1. Calculate the sensitivity $\Delta count_P$ and $\Delta max - salary - at - Berkeley$ for the query functions. Calculate the variances of the distributions from which noise is added in order to achieve ϵ-differential privacy with $\epsilon = 0.00001$ and $\epsilon = 0.001$ for both query functions. Now let's try to understand these results qualitatively. For the "same level of privacy", which function requires "more noise" to be added? For a fixed function, how does the "noise distribution change" in order to achieve "higher levels of privacy"? Your answers should make precise all the terms in quotes.

2. Let $D_1 = \{A_P; B; C_P\}$ and $D_2 = \{A_P; B; C_P; D_P\}$ be two neighboring databases where an element of the form X_P has the property P (e.g., A_P has property P, but B does not). Calculate the values of $count_P(D_1)$ and $count_P(D_2)$. Recall that $\kappa_f(D) = f(D) + Y$ where $Y \sim Laplace(\Delta f/\epsilon)$, i.e., Y is a random variable that follows a Laplace distribution with mean 0 and variance $2(\Delta f/\epsilon)^2$. Calculate $Pr(\kappa_{count_P}(D1) \in (2.3, 2.7])$ and $Pr(\kappa_{count_P}(D2) \in (2.3, 2.7])$ and verify that their ratio is less than e^ϵ. Notice that you are verifying that the differential privacy definition is satisfied for $S = (2.3, 2.7]$. You can use $\epsilon = 0.001$ in your calculations.
   Hint: Look up the Wikipedia page for Laplace distribution.

Problem 3 (50 marks)

Additional files required for this homework are available at `https://www.ece.cmu.edu/~ece734/homework/hw4-files.zip`.

In this question, we are going to work through how deanonymization works on databases. We will work with a subset of the original Netflix database. You are given a set of 15 movies in the folder *movies*. The identifiers for these movies are [03124, 06315, 07242, 16944, 17113, 10935, 11977, 03276, 14199, 08191, 06004, 01292, 15267, 03768, 02137]. The *movies* folder contains csv files for each movie. Each line of the csv file has three entries: a user id, the date of rating, and the rating provided.

Learning Objectives: The problems in the part are based on the paper 'Provable Deanonymization of Large Datasets with Sparse Dimensions' [1]. We will refer to the paper through all problems in this part. In Problem 1, we will perform an attack along the lines of the original Netflix-IMDB deanonymization attack. In particular, we will learn how to identify a user by utilizing noisy and incomplete auxiliary information. In Problem 2, we will analyze a theorem that enables such attacks (called Isolation Attacks) under certain assumptions about the dataset and the auxiliary information. In Problem 3, we will connect the findings from the above two problems. Specifically, we will empirically calculate the fraction of users for whom the assumptions in the above theorem hold. In order to do so, we will use the attack technique used in Problem 1.

Starter Code: You may use the provided starter code (in Python) for this homework. The script reads each file from the movies folder and populates the database *db*. *db* is a Python dictionary. Dictionaries consist of pairs (called items) of keys and their corresponding values. To brush up your knowledge of the Python dictionary data-structure, please view this tutorial[2]. Each element of *db* is the tuple ⟨*user-id, movie-dict*⟩. *movie-dict* is also a dictionary representing the user's ratings, with each item being the tuple ⟨*movie-id, rating*⟩).

It is not compulsory that you use the starter code provided. You can use any programming language of your choice (among C/C++/Java/Python).

3A: An attack (15 marks)

Starter code for this problem is provided in `link.py`.

You are given the auxiliary information for one user. The auxiliary information contains noisy ratings given by the user for 12 of the 15 movies. You can think of these being perturbed ratings

[1] `http://www.andrew.cmu.edu/user/divyasha/dss-post12.pdf`
[2] `http://www.tutorialspoint.com/python/python_dictionary.htm`

given by a user on IMDB. This auxiliary information is provided in Table 1 and in the variable `aux` in `link.py`. You, as the attacker, want to identify the user id for whom the auxiliary information is provided.

movie	rating	movie	rating	movie	rating	movie	rating
14199	4.5	17113	4.2	06315	4.0	01292	3.3
11977	4.2	15267	4.2	08191	3.8	16944	4.2
07242	3.9	06004	3.9	03768	3.5	03124	3.5

Table 1: Auxiliary information for the target user.

(a) Using Definition 4 of the paper, complete the function `compute_weights()` and compute the weights of each movie. Tabulate the weights obtained for each movie. This should be a table with 15 movie-ids and their corresponding weights.

(b) How many users are present in the database? Using Definition 7 in the paper, complete the function `score()` and compute the scores of the auxiliary information with respect to every user's ratings in the database.

What is the highest score? What is the second highest score?

(c) What is the user-id of the user with the highest score? Write out the ratings of this user from the database, and verify if they are similar to the ratings in the auxiliary information.

3B: Verify correctness of linking using Theorem 2 (20 marks)

Starter code for this problem is provided in `theorem.py`.

In the previous problem, we were able to link the auxiliary information with a record in the anonymized database. But, we are not sure if the link we found is correct. Theorem 2 on Page 8 of the paper gives guarantees about the correctness of the linking, given two assumptions:

- The auxiliary information is (m, γ)-perturbed.
- The eccentricity measure (Definition 8 in the paper) is greater than γM where $M = \frac{\sum_{i \in supp(aux_y)} w_i}{|supp(aux_y)|}$ is the scaled sum of weights of attributes in aux_y

(a) Given that the auxiliary information in Table 1 is generated from using $m = 12$ and $\gamma = 0.1$, check if the eccentricity assumption holds. What does this say about your attack in $3A$? Were you able to correctly link the auxiliary information with the deanonymized database?

We will now find for what proportion of the records, does the eccentricity assumption hold for each of $\gamma \in \{0.1, 0.2\}$ and $m \in \{8, 10\}$. For each record in the database, create auxiliary information using the function `create_aux()`. This function is provided in the starter code. It takes as input a record (an element of *db*), and parameters m and *gamma* and returns a random subset of m movies from the set of movies the user has seen, and adds noise bounded by *gamma* to each rating. To create (m, γ)-perturbed auxiliary information for a user, the user must have seen at least m movies. So, you must filter out all users who have rated less than m movies. The `create_aux()`

function checks for requirement and throws an error if this condition is not met. `create_aux()` adds noise to the ratings in the auxiliary information using the function `add_noise()`. If you are using the starter code, then do not change the `random.seed()` - your results will be verified based on the random numbers generated using that seed. If you are not using the starter code, you can add γ-bounded noise in the same manner as in the starter script, however we expect the random number generator to be different. In that case, your implementation will be manually checked for correctness.

(b) For each of $m \in \{8, 10\}$, how many user-records remain after filtering?

To compute the proportion of user records for which the assumptions hold, we will carry out the following. For each user-record in the database:

1. Create auxiliary information `aux` using m and γ.
2. Check if the eccentricity assumption holds for `aux` and the database.

Since there is some randomness in creating each `aux`, we will repeat the above process several times, and compute the average proportion over several runs.

(c) Compute the proportion (averaged over 5 runs) of user-records for which the eccentricity assumption holds for each of $\gamma \in \{0.1, 0.2\}$ and $m \in \{8, 10\}$. Plot the proportions in a bar-diagram along the lines of Figure 1 on page 15 of the paper. The figure should have 4 bars. (You do not have to worry about t.) REMEMBER: the proportion is over the number of records remaining after filtering.

3C: Proving a modified Theorem 2 (15 marks)

Read Theorem 2 on Page 8 from the paper. Consider a different definition for perturbation:

Definition 1 ((m, ζ)-perturbed auxiliary information) *Auxiliary information about record $y \in D$, denoted by aux_y, contains perturbed values of m non-null attributes sampled from attributes in record y. aux_y is defined to be (m, ζ)-perturbed if $\forall i \in supp(aux_y).T(y(i), aux_y(i)) \geq \zeta$ where $0 \leq \zeta \leq 1$.*

And a different definition of weight.

Definition 2 (α-weight of an attribute) *The scaled weight of an attribute i is denoted by w_i^* and is defined as $w_i^* = \frac{\alpha}{\log_2 |supp(i)|}$.*

Now, note a variant of Theorem 2, using modified definitions of perturbation and weight:

Theorem *Let y denote the target record from given database D. Let aux_y denote (m, ζ)-perturbed auxiliary information about record y. If the eccentricity measure $e(aux_y, D) > \beta M$ where $M = \frac{\sum_{i \in supp(aux_y)} w*_i}{|supp(aux_y)|}$ is the scaled sum of α-weights of attributes in aux_y, then*

1. $\max_{r \in D}(Score_w(aux_y, r)) = Score_w(aux_y, y)$.
2. *Additionally, if only one record has maximum score value $= Score_w(aux_y, y)$, then the record returned by the algorithm o is the same as target record y.*

The proof of the original Theorem 2 is done by contradiction. Prove the modified Theorem 2 along the same lines. In doing so, find the value of β in the lower bound for eccentricity so that the theorem holds.

Submission

You have to submit three files:

1. Merge all the written parts into a single pdf file ⟨your_andrew_id⟩_HW4.pdf.
2. Rename the program file (.c/.cpp/.java/.py) you used for 3A as ⟨your_andrew_id⟩_link.⟨extension⟩.
3. Rename the program file (.c/.cpp/.java/.py) you used for 3B as ⟨your_andrew_id⟩_theorem.⟨extension⟩.

Zip these three files into ⟨your_andrew_id⟩_HW4.zip and submit the zip file on BlackBoard.