TEL AVIV UNIVERSITY
Department of Computer Science
0368.3239 – Foundations of Data Mining
Fall Semester, 2017/2018

Homework 3, December 19, 2017

- **Due on January 2 23:59 IST. Each question has the same weight.**
- **Submission instructions: We are using `https://gradescope.com`. Gradescope entry code for the course is: MYVDZ5**
  Please prepare a PDF file with each problem starting on a new page. When uploading, you will need to indicate locations of each problem/section.
- **You may consult any sources or people but you must write and submit the solution yourself and state your collaborators.**

1. Let A be a matrix with n rows and d columns, assume $n \geq d$. Let $U\Sigma V^T$ be the Singular Value Decomposition of A as defined in class, and let $U_k\Sigma_k V_k^T$ be its truncation to rank k as defined in class.

Prove that

$$||A - A_k||_2 = \sigma_{k+1}(A) = \min_B ||A - B||_2$$

where the minimum is over all $n \times d$ matrices B of rank k.

2. Let A be a matrix with n rows and d columns, assume $n \geq d$.

a. Characterize the vector x which minimizes $||Ax - b||_2$ in terms of the SVD of A. Prove your answer.

b. What is the value of $\min_{x \in \mathbb{R}^d} ||Ax - b||_2$?

3. We have a long file of names (each name may appear multiple times) and we want to privately publish the name with maximum frequency. We do it by adding to the frequency n_i of each name i, a random variable N_i drawn from a $Lap(1/\epsilon)$ distribution and publishing the name j with the largest noisy frequency $n_j + N_j$. Prove that this mechanism is $(2\epsilon, 0)$-differentially private.

4. Given a database $S = \{s_1, s_2, \ldots, s_n\}$, all values integer and distinct between 1 and T. we seek to give an $(\epsilon, 0)$-differential privacy preserving approximation to the median element in the database. Two databases are at distance one if they differ in exactly one element.

The utility of an "approximate median" r for S is defined as follows: Let Q be the set of all sets R with values between 1 and T , such that the median of R is equal to r. For all $R \in Q$, let $\Delta(R)$

be the size of the symmetric difference between S and R (The number of elements needed to be added/deleted so as to change S to R or R to S). Let

$$\Delta = \min_{R \in Q} \Delta(R).$$

The utility of an approximate median r, $u(r) = -\Delta$.

a. (5 points) What is the sensitivity of this utility (Δ_u as defined in class and in Dwork and Roth)

b. (5 points) The exponential mechanism outputs an integer $r \in 1, \ldots, T$ with probability proportional to $e^{(\epsilon/2)u(r)}$. Is it the case that the expected (absolute value) of the difference between the median and the approximate median produced by the exponential mechanism small? Prove so or give a counter example.

c. (5 points) What about simply returning the i'th largest item from S where $i = \lfloor n/2 + Lap(1/\epsilon) \rfloor$. (Round up to 1 if below 1, round down to n if above n). For what value of ϵ is this mechanism $(\epsilon, 0)$-differentially private? (Could be that nothing can be done).

d. (5 points) Can you devise a locally differentially private mechanism for the approximate median problem? Hints:

- This will be an interactive protocol where the center interacts multiple times with all the individuals:
- The protocol has $O(\log T)$ rounds:
  - A predicate is broadcast $P_i : \{1, \ldots, T\} \mapsto \{0, 1\}$.
  - Count (approximately) the number of times the predicate gives true using randomized response.

Fill in the missing parts of this description above so as to get an approximate median. What utility will one get for the result? **Remark:** There is also a non-interactive locally differentially private mechanism, which is more complex. We are asking about the interactive version.

5. A 2-d tree T is a binary tree that represents a set P of points in the plane as follows. Every leaf stores a point of P. Every non-leaf node v at an even distance from the root corresponds to a horizontal line $\ell(v)$ such that $\lfloor k/2 \rfloor$ of the k points in v's subtree are above $\ell(v)$ and are stored at the left subtree of v and $\lceil k/2 \rceil$ of the of the k points in v's subtree are below $\ell(v)$ and are stored at the right subtree of v. Similarly, every non-leaf node v at an odd distance from the root corresponds to a vertical line $\ell(v)$ such that $\lfloor k/2 \rfloor$ of the k points in v's subtree are to the left of $\ell(v)$ and are stored at the left subtree of v and $\lceil k/2 \rceil$ of the of the k points in v's subtree are to the right of $\ell(v)$ and are stored at the right subtree of v.

a. (2 points) Prove that the leaves of a subtree rooted by a node v correspond to the points of P that are contained in a rectangle $r(v)$. Specify the boundaries of this rectangle.

We can use 2-d tree storing a set of points P to find all points in some query rectangle R as follows. We start at the root and for each internal node v which we visit we perform the following. If the rectangle $r(u)$ corresponding to the right child u of v is contained in R then we report all points in the subtree of u. Similarly, if the rectangle $r(w)$ corresponding to the left child w of v is contained in R then we report all points in the subtree of w.
If R intersects $r(u)$ but does not contain it then we continue the search in u. Similarly, if R intersects $r(w)$ but does not contain it then we continue the search in w. When we reach a leaf z then we report the point stored at z if it is contained in R.

b. (6 points) Prove that the running time of this algorithm is $O(\sqrt{n} + k)$ where n is the size of P and k is the number of points that are contained in $\mathbb{R}$. (Hint. Count separately the sizes of the subtrees of nodes v where $r(v) \subseteq R$ and all the other nodes which the search visits.)

c. (6 points) Describe a generalization of the 2-d tree to a k-d tree storing a set of points P in $\mathbb{R}^k$. Generalize also the algorithm for reporting all points in a box R in $\mathbb{R}^k$.

d. (6 points) Describe an algorithm that uses a k-d tree to find the nearest neighbor of a point $x \in \mathbb{R}^k$. What is the worst case running time of your algorithm ? Prove your answer.