15-859 Algorithms for Big Data — Fall 2019

Problem Set 3

Due: Friday, November 1, 11:59pm. Please email Raj your solutions.

Please see the following link for collaboration and other homework policies:
`http://www.cs.cmu.edu/afs/cs/user/dwoodruf/www/teaching/15859-fall19/grading.pdf`

Problem 1: Fun with Cauchy Random Variables and ℓ_1 (25 points) We saw in class that given an $n \times d$ matrix A, $n \geq d$, it was possible to choose an $O(d \log d) \times n$ random matrix R such that with probability at least $9/10$, simultaneously for all x,

$$\|Ax\|_1 \leq \|RAx\|_1 = O(d \log d)\|Ax\|_1.$$

The matrix R was a matrix of i.i.d. Cauchy random variables, each scaled by $1/r$, where $r = O(d \log d)$ is the number of rows of R.

Suppose we instead choose R to have a larger number of rows, but still require R to be a matrix of i.i.d. Cauchy random variables. You can choose to scale R by any number you would like. Show that if $r = 2^{2^{O(d)}}$ and for an appropriate scaling, with probability at least $9/10$, simultaneously for all x,

$$\|Ax\|_1 \leq \|RAx\|_1 = O(1)\|Ax\|_1. \tag{1}$$

HINT: Look at what R does to a fixed vector Ax and with what probability. It may be helpful to note that for a Cauchy random variable C and for any real number $t > 1$,

$$\Pr[|C| \in (t, 2t]] = \Theta(1/t).$$

You may want to understand the number of entries of RAx with absolute value in the range $S_i = [2^i, 2^{i+1})$ for each value of i, and how concentrated this number is around its expectation.

You may also want to separately analyze very large values of i. For small i, when S_i has a large number of expected members, you should be able to show good concentration with large probability. For larger i, meaning when $\mathbb{E}|S_i|$ is smaller, you will have to upper bound the contribution of these ranges with lower (but still sufficient) probability. However, you may only need to consider a few such *large* sets (at what point i can we condition on $|S_j| = 0$ for all $j > i$?).

Finally, after you understand what R does to a fixed vector Ax, you should apply a net argument. Remember that to apply a net arguement, your failure probability for a fixed Ax must be $2^{-O(d)} = O(\log^{-1}(r))$, so all events you condition on should hold with at least this probability. Note: you may get slightly lower probability for some events, such as $O(\frac{\log \log r}{\log r})$, but this can still be bounded by $2^{O(-d)}$.

Problem 2: Data Stream Algorithms and Lower Bounds (25 points) Recall in the data stream model there is an underlying vector $x \in \{-M, -M+1, \ldots, M\}^n$, for some integer $M \leq \text{poly}(n)$, which is initialized to 0^n, and which undergoes a long sequence of additive updates of the form $x_i \leftarrow x_i + \Delta_j$ to its coordinates. We are promised at all times during the stream that $x \in \{-M, \ldots, M\}^n$.

(1) (15 points) Design a data stream algorithm which at the end of the stream, outputs a number Y such that with probability at least $9/10$, it holds that $(1-\epsilon)(2\|x\|_2^2 - \|x\|_1) \leq Y \leq (1+\epsilon)(2\|x\|_2^2 - \|x\|_1)$. You can assume you have random access to a read-only string of poly(n) random bits, and all the randomness of the algorithm comes from this tape, so the $9/10$ probability is taken over the choice of this random tape. Here, by random access, we mean you can access any desired entry of this tape at any time during the stream. Your algorithm should use $O((\log n)/\epsilon^2)$ bits of memory.

HINT: If you use a continuous distribution in your algorithm you may need to discretize the entries to get the desired memory, and argue what happens when you discretize.

(2) (10 points) In the promise multi-player set disjointness communication problem, there are t players $P^1, \ldots, P^t$, where the i-th player P^i holds a set $S_i \subseteq \{1, 2, \ldots, n\}$, and the promise is that we are in one of two cases:

1. for all $i \neq j$, $S_i \cap S_j = \emptyset$, or
2. there exists a unique element k for which $k \in S_i$ for all $i = 1, 2, \ldots, t$, and for all $k' \in \{1, 2, \ldots, n\}$ with $k' \neq k$, we have that k' occurs in at most a single set S_i.

Suppose that the players are arranged in the coordinator model of communication that we saw in class, where each player can send messages to and from a centralized coordinator. It is known, for any setting of the parameters n and t, that any randomized protocol for distinguishing which of the two cases we are in with probability at least $9/10$, requires $\Omega(n/t)$ bits of total communication between the players and the coordinator. Note that this lower bound holds for any number of rounds of communication.

Now consider the p-norm estimation problem in a data stream for $p > 2$. Here we would like to output an estimate Y for which $\|x\|_p^p \leq Y \leq 2\|x\|_p^p$, with probability at least $9/10$. Recall that x is an underlying vector, initialized to 0^n, and at all times in the stream $x \in \{-M, -M+1, \ldots, M\}^n$, for some $M \leq \text{poly}(n)$. Using the above, show that any streaming algorithm for solving this problem requires $\Omega(n^{1-2/p})$ bits of memory.