15-859 Algorithms for Big Data — Fall 2017
Problem Set 3 Solutions

Problem 1: Entrywise-ℓ_1 Low Rank Approximation

(1) Let $T = r \cdot S$. By the 1-stability of the Cauchy distribution, each entry of the $r \times n$ matrix TA is a Cauchy random variable, where the entries in the j-th column are independent and scaled by $\|A_j\|_1$, where A_j is the j-th column of A. Let $\mathcal{E}_{i,j}$ be the event that $(TA)_{i,j} \leq n^3\|A_j\|_1$, and let $\mathcal{E} = \cap_{i,j}\mathcal{E}_{i,j}$. Then $\Pr[\mathcal{E}_{i,j}] \geq 1 - O(1/n^3)$, and using that $r \leq n$, by a union bound $\Pr[\mathcal{E}] \geq 1 - O(1/n)$. Repeating the argument shown in class, $\mathbf{E}[|(TA)_{i,j}| \mid \mathcal{E}_{i,j}] = \Theta(\log n)\|A_j\|_1$. Also, repeating the argument shown in class, $\mathbf{E}[|(TA)_{i,j}| \mid \mathcal{E}] = \Theta(\log n)\|A_j\|_1$. Consequently, $\mathbf{E}[\|TA\|_1 \mid \mathcal{E}] = \Theta(r \log n)\|A\|_1$, and so by a Markov bound, with probability at least 9/10, $\|TA\|_1 = O(r \log n)\|A\|_1$. It follows that with probability at least 9/10, $\|SA\|_1 = O(\log n)\|A\|_1$.

(2) The original version of the problem set asked you to just show an $O(k \log k + \log n)$ bound, and you will receive full credit if you show this bound. For this part of the problem it is possible to achieve an $O(\log n)$ approximation, so we'll show the tighter approximation factor.

If V' is the minimizer to $\min_{V'} \|SUV' - SA\|_1$, then $\|UV' - A\|_1 \leq \|UV' - UV^*\|_1 + \|UV^* - A\|_1$ by the triangle inequality. Also, $\|UV' - UV^*\|_1 \leq \|S(UV' - UV^*)\|_1$ by the property of S, and $\|S(UV' - UV^*)\|_1 \leq \|SUV' - SA\|_1 + \|SUV^* - SA\|_1$ again by the triangle inequality. Then, $\|SUV' - SA\|_1 \leq \|SUV^* - SA\|_1$ since V' is the minimizer to $\min_{V'} \|SUV' - SA\|_1$, and so $\|S(UV' - UV^*)\|_1 \leq 2\|SUV^* - SA\|_1 = O(\log d)\|UV^* - A\|_1$ by the previous part. Putting this all together, $\|UV' - A\|_1 = O(\log d)\|UV^* - A\|_1$.

(3) Let V'' be the minimizer to $\min_V \|S(UV - A)\|_{1,2}$, as suggested by the hint. Notice that we can solve for the columns of V'' independently. Also notice that for any individual column V_i'', we have $V_i'' = (SU)^-(SA_i)$ by the normal equations applied to that column, since this is just a least squares problem. Consequently, $V'' = (SU)^- SA$, and is thus in the row span of SA. Since S has only s rows, for any vector V_i,

$$\frac{1}{\sqrt{s}}\|S(UV_i - A_i)\|_1 \leq \|S(UV_i - A_i)\|_2 \leq \|S(UV_i - A_i)\|_1.$$

It follows that $\|S(UV_i'' - A_i)\|_1 \leq \sqrt{s}\|S(UV_i'' - A_i)\|_2 \leq \sqrt{s}\|S(UV_i' - A)\|_2 \leq \sqrt{s}\|S(UV_i' - A)\|_1$, where the first inequality is as above, the second inequality uses that V'' is the minimizer with respect to the $\|\cdot\|_{1,2}$ norm, and the third inequality uses that the 2-norm is at most the 1-norm. Here V' is the minimizer to $\min_V \|S(UV - A)\|_1$, as in the previous part. Consequently, $\|S(UV'' - A)\|_1 \leq \sqrt{s}\min_V \|S(UV - A)\|_1$.

(4) Notice that we have for any Y, X that

$$\begin{aligned}\frac{\|T_L ARYXSAT_R - T_L AT_R\|_1}{k \cdot \text{poly}(\log k)} &\leq \|T_L ARYXSAT_R - T_L AT_R\|_F \\ &\leq \|T_L ARYXSAT_R - T_L AT_R\|_1,\end{aligned}$$

which follows by thinking of $T_L ARYXSAT_R - T_L AT_R$ as a length-$k^2\text{poly}(\log k)$ dimensional vector and using the relationship between the 1-norm and 2-norm for such vectors. Consequently, by minimizing with respect to the Frobenius norm, we have that $\beta \leq k \cdot \text{poly}(\log k)$. Hence, using that T_L and T_R are affine embeddings for ℓ_1, overall our approximation ratio is $\text{poly}(k \log n)$.

Problem 2: Estimating Quantities in a Stream

(1) Let S be a CountSketch matrix with $O(1/\epsilon^2)$ rows. As discussed in class, S can be represented using $O(\log n)$ bits by storing its hash function h, which can be 2-wise independent, and sign function σ, which can be 4-wise independent. Problem 3 on the first homework showed that for any fixed vector x (in fact for any matrix), we have $\|Sx\|_2^2 = (1 \pm \epsilon)\|x\|_2^2$ with probability at least 9/10. We can maintain Sx in the streaming model using $O((\log n)/\epsilon^2)$ bits of space. We can also maintain μ in the streaming model with $O(\log n)$ bits of space, since this just amounts to maintaining the single inner product of x with the vector $(1/n, 1/n, \ldots, 1/n)$. At the end of the stream, we compute $S \cdot (\mu \cdot 1^n)$, where $(\mu \cdot 1^n)$ is the vector with μ in every coordinate. We can then compute $Sx - S \cdot (\mu 1^n)$, and we have by the above that

$$\frac{1}{n}\|Sx - S \cdot (\mu 1^n)\|_2^2 = (1 \pm \epsilon)\frac{1}{n}\|x - (\mu 1^n)\|_2^2 = (1 \pm \epsilon)\frac{1}{n}\sum_i (x_i - \mu)^2 = (1 \pm \epsilon)v,$$

which holds with probability at least 9/10.

(2) Note that $(x_i^2 - 8x_i + 16) = (x_i - 4)^2$. Hence, as in the previous part, we can output an $\tilde{f}$ for which $f(x)/2 \leq \tilde{f} \leq 3f(x)/2$ for the function $f(x) = \sum_{i=1}^n (x_i^2 - 8x_i + 16)$ using $O(\log n)$ bits of space by maintaining $S \cdot x$ for a CountSketch matrix S with $O(1)$ rows, then computing $S \cdot (4 \cdot 1^n)$ at the end of the stream, and outputting $\|Sx - S \cdot (4 \cdot 1^n)\|_2^2 = (1 \pm 1/2)\|x - (4 \cdot 1^n)\|_2^2 = (1 \pm 1/2)\sum_i (x_i - 4)^2$, which holds with probability at least $9/10 > 2/3$.

For the other function $f(x) = \sum_{i=1}^n (x_i^2 - 10x_i + 16)$, we have $f(x) = \sum_{i=1}^n (x_i - 8)(x_i - 2)$, and we claim that any randomized algorithm which succeeds with probability at least 2/3 in outputting an $\tilde{f}$ with $f(x)/2 \leq \tilde{f} \leq 3f(x)/2$ requires $\Omega(n)$ bits of space. Suppose we are presented a stream which is an instance of problem $\mathcal{P}$. For each stream update of the form $x_i \leftarrow x_i + \Delta_j$, we replace it with $x_i \leftarrow x_i + 6\Delta_j$. Finally, at the end of the stream, we include the updates $x_i \leftarrow x_i + 2$ for $i = 1, 2, \ldots, n$. Then if the original instance x to problem $\mathcal{P}$ had the property that all $x_i \in \{0, 1\}$ at the end of the stream, the vector x produced by this transformed stream has the property that all $x_i \in \{2, 8\}$ at the end of the stream. Note that in this case, we have $f(x) = \sum_{i=1}^n (x_i - 8)(x_i - 2) = 0$. On the other hand, if the original x to problem $\mathcal{P}$ had a coordinate $x_i \notin \{0, 1\}$, then $6x_i + 2 \notin \{2, 8\}$, and so $f(x) \neq 0$. Any number $\tilde{f}$ for which $f(x)/2 \leq \tilde{f} \leq 3f(x)/2$ can distinguish these two cases, and thus, if we feed the updates in this transformed stream to an algorithm for outputting such an $\tilde{f}$ with probability at least 2/3, we can solve problem $\mathcal{P}$ with probability at least $2/3$. Hence, any such algorithm requires $\Omega(n)$ bits of space.