CS 224 Advanced Algorithms — Spring 2017
Problem Set 6

Due: 11:59pm, Wednesday, April 5th

Submit solutions to Canvas, one PDF per problem:
`https://canvas.harvard.edu/courses/21996`

Solution max page limits: 2 pages for each of problems 1 and 2 (ignoring code included for problem 1)

See homework policy at `http://people.seas.harvard.edu/~cs224/spring17/hmwk.html`

Problem 1: This problem will explore Locality-Sensitive Hashing (LSH) schemes for ℓ_1 and ℓ_2 metrics. Recall the ℓ_p norm of a vector x is $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$.

Also recall that in the (c, r)-*NN problem*, we have to pre-process a database of n points in some metric space subject to queries. A query is specified by a point q in that metric space. If q is within distance r of some database point p, we must return a (possibly different) database point within distance cr of q with failure probability at most f.

(a) (6 points) Suppose there is a data structure for (c, r)-NN in $[0, ar]^d$ under the ℓ_1 norm that uses space $S(n, f)$, has query time $Q(n, f)$, and has failure probability f. Then show there exists a data structure for the same problem over ℓ_1^d, with asymptotically the same query time and space bounds, and failure probability at most $f + 1/a$. You should assume that $S(n, f) \geq S(a, f) + S(b, f)$ for any nonnegative integers $a + b = n$. You should also assume $Q(n, f) \geq d$, and $Q(n, f)$ monotonically increases with n. **Hint:** randomly shift a grid.

(b) (2 points) Based on the Hamming metric LSH scheme in lecture, conclude that for any $0 < \delta < 1$ there's a solution to $(c(1 + \delta), r)$-NN in ℓ_1^d with failure probability 1/5 using space $O(d'n + n^{1+\rho})$ and query time $O(d'n^\rho)$ for $\rho = 1/c$ and $d' = O(d^2/\delta)$.

(c) (7 points) It is known that if X is any set of vectors, there is always an efficiently computable map $f : X \to X'$ such that for all $x, y \in X$, $\|f(x) - f(y)\|_1 = \|x - y\|_2$. This leads to (c, r)-NN schemes for ℓ_2 with $\rho \approx 1/c$ (by first embedding into ℓ_1 then using (b)). Another approach is to use "SimHash". In SimHash, we imagine that all points in the database, as well as the query point, lie on the unit sphere (i.e. have Euclidean norm 1). We pick a random gaussian vector g (or a random point on the sphere would work too), and we hash x to $sign(\langle g, x\rangle)$. This is similar to the rounding step in the Goemans-Williamson SDP-based algorithm for MaxCut. Consider $\rho_r = \rho_r(c)$ to be the "ρ" for (c, r)-NN (so $\rho = \sup_r \rho_r$). Recall $\rho_r = \log(1/p_1)/\log(1/p_2)$, where p_1 is the probability of hash collision for points within distance r, and p_2 is the probability of collision for points at distance at least cr. Write down an analytic expression, in terms of c, r, for $\rho_r(c)$ with SimHash. Then run some experiment, e.g. using code, to estimate ρ for $c = 2^j$ for $j = 1, \ldots, 15$. Does ρ asymptotically look like $\Theta(1/c)$? $\Theta(1/c^2)$? Something else? And, empirically, what does the constant in the $\Theta(\cdot)$ look like for large c? Also, in class it was mentioned that in recent data-dependent LSH works, performance is better in data-oblivious LSH when points are "random". It is true, though beyond the scope of this course to prove, that random points on the sphere in dimension d for large d have pairwise distances close to maximum possible (i.e. $\sqrt{2}$) with large probability. Thus we may consider the case that "far points" are random, and have distance $cr \approx \sqrt{2}$. Again, empirically, what is the behavior of ρ_r as a function of c when $r = \sqrt{2}/c$? Is it $\Theta(1/c)$ or $\Theta(1/c^2)$, or something else? Again, what does the constant in the $\Theta(\cdot)$ seem to be? Include your code. I recommend using the `listings` and `color` packages in LaTeX; see for example `http://tex.stackexchange.com/questions/83882/how-to-highlight-python-syntax-in-latex-listings-lstinputlistings-command`. (see the macros at the beginning of this problem set's source code, as well as the commented out command on the following line)

(d) (Bonus, 5 points) Perform a rigorous, mathematical analysis for both ρ and $\rho_{\sqrt{2}/c}$ in (c). Try to understand both these quantities up to $1 + o(1)$ factors as c grows.

Problem 2: In class we saw an idealized algorithm for estimating the number of distinct elements in a stream. Here we will develop an idealized algorithm for the turnstile version of this problem: $x \in \mathbb{R}^n$ receives updates of the form "$x_i \leftarrow x_i + \Delta$" (the increments Δ may be positive or negative), and we must answer `query`() with a value $\tilde{L}_0$ such that

$$\mathbb{P}(|\tilde{L}_0 - L_0| > \varepsilon L_0) < 1/3. \tag{1}$$

Here L_0 is defined to be the support size of x, i.e. $L_0 = |\{i : x_i \neq 0\}|$. Note the number of distinct elements is just L_0 when all the increments have $\Delta = 1$ (whenever we see $i \in \{1, \ldots, n\}$ in the stream, we interpret it as the update $x_i \leftarrow x_i + 1$).

Suppose we are given a value K, which is a power of 2, in the beginning of the stream and are promised that during a query L_0 will always be in the interval $[K, CK]$ for some constant $C \geq 1$ told to us. I propose the following first step toward obtaining an estimate satisfying (1). Say n is a power of 2 (else round it up). We pick a totally random hash function $h : [n] \to [n]$. We then define $\ell(i)$ to be the index of the least significant bit of $h(i)$, so that $\ell(i) = j$ with probability $1/2^{j+1}$. Then when we see an update to the stream "$x_i \leftarrow x_i + \Delta$", if $\ell(i) \neq \log_2 K$, we *ignore the update*! Otherwise, we include the update in the stream. In this way, we obtain a randomly sampled substream, containing only updates to a randomly sampled subset I of $[n]$.

(a) (2 points) Give an exact expression for $\mathbb{P}(x_I \neq 0)$, in terms of both K and L_0. Here x_I is the $|I|$-dimensional vector obtained by projecting x to only the coordinates in I. Using that $(1 - 1/t)^t \approx 1/e$ for large t, explain why your calculated expression is bounded away from both 0 and 1.

(b) (2 points) Give a randomized algorithm $\mathcal{A}$ for testing whether a vector x being updated in the turnstile streaming model is identically zero. Your algorithm should use $O(\log(1/\delta))$ words of memory to have failure probability at most δ.

(c) (4 points) Combine (a) and (b) to obtain an algorithm for estimating L_0 as in (1). Your algorithm should use space $O(poly(\varepsilon^{-1} \log n))$ when given the value K as discussed above.

(d) (2 points) In reality, you don't know such a K. Show how to remove the assumption that you are given such a K while still using space $O(poly(\varepsilon^{-1} \log n))$.