Fall 2014, CS 598: Algorithms for Big Data

Homework 4
Due: 11/20/2014

Instructions and Policy: Each student should write up their own solutions independently. You need to indicate the names of the people you discussed a problem with; ideally you should discuss with no more than two other people.

Solve as many problems as you can.

You need to typeset your solutions in Latex. Please write clearly and concisely - clarity and brevity will be rewarded. Refer to known facts as necessary. Your job is to convince me that you know the solution, as quickly as possible.

Problem 1 Recall the JL Lemma where we pick a random $m \times n$ matrix Π and show that for $m = O(1/\epsilon^2)$, with at least $2/3$ probability,

$$(1-\epsilon)\|x\|_2^2 \leq \|\Pi x\|_2^2 \leq (1+\epsilon)\|x\|_2^2. \tag{1}$$

(a) (5 points) Imagine picking Π as follows: for each $i \in \{1, \ldots, n\}$ we pick a uniformly random number $h_i \in \{1, \ldots, m\}$. We then set $\Pi_{h_i,i} = \pm 1$ for each $i \in \{1, \ldots, n\}$ (the sign is chosen uniformly at random from $\{-1, 1\}$), and all other entries of Π are set to 0. This Π has the advantage that in turnstile streams, we can process updates in constant time. Show that using this Π still satisfies the conditions of Equation 1 with $2/3$ probability for $m = O(1/\epsilon^2)$.

(a) (5 points) Show that the matrix Π from Problem 3(a) can be specified using $O(\log n)$ bits such that Equation 1 still holds with at least $2/3$ probability, and so that given any $i \in \{1, \ldots, n\}$, $\Pi_{h_i,i}$ and h_i can both be calculated in constant time. *Hint:* Use limited independence hash functions to generate the h_i.

Problem 2 In class we saw an LSH scheme for nearest neighbor search for n binary strings of length d in the hamming distance metric. The scheme was based on a decision version where for a given r the data structure would be able to answer with good probability whether there is a point in the data base with distance at most r from q or whether every point is at least $(1+\epsilon)r$. The final data structure is composed of $O(\log d/\epsilon)$ data structures for different values of r. Do we need this reduction to the decision version? Read Charikar's paper on similarity search for a variant of the basic scheme that avoids this in a simple way. Describe and analyze the scheme in your own words.

Problem 3 In class we saw the frequent directions algorithms for low rank approximation with parameter k. The algorithm computes n SVDs of $\ell \times d$ matrices where $\ell = k(1 + 1/\epsilon)$

each of which takes $O(d\ell^2)$ time for a total time of $O(ndk^2/\epsilon^2)$. We would like to improve the run time to $O(ndk/\epsilon)$. Consider some constant $c > 1$ and now we will use matrix Q with $c\ell$ rows but we will batch every $(c-1)\ell$ rows of A and compute an SVD only once for each batch. Show that this scheme gives $(1+\epsilon)$-approximation as before but the running time is now reduced to $O(\frac{c^2}{c-1}nd\ell)$.