Problem Set 5 CS265, Autumn 2022

Due: Friday 11/4 at 11:59 pm on Gradescope

Please follow the homework policies on the course website.

---

1. **(14 pt.) [Another way to sketch sparse vectors.]** Suppose that A is an list of length n, containing elements from a large universe $\mathcal{U}$. Our goal is to estimate the frequencies of each element in $\mathcal{U}$: that is, for $x \in \mathcal{U}$, how often does x appear in A?

   The catch is that A is too big to look at all at once. Instead, we see the elements of A one at a time: $A[0], A[1], A[2], \ldots$. Unfortunately, $\mathcal{U}$ is also really big, so we can't just keep a count of how often we see each element.

   In this problem, we'll see a construction of a randomized data structure that will keep a "sketch" of the list A, use small space, and will be able to efficiently answer queries of the form "approximately how often did x occur in A"?

   Specifically, our goal is the following: we would like a (small-space) data structure, which supports operations `update`(x) and `count`(x). The `update` function inserts an item $x \in \mathcal{U}$ into the data structure. The `count` function should have the following guarantee, for some $\delta, \epsilon > 0$. After calling `update` n times, `count`(x) should satisfy

   $$C_x \leq \texttt{count}(x) \leq C_x + \epsilon n \tag{1}$$

   with probability at least $1 - \delta$, where C_x is the true count of x in A.

   (a) **(3 pt.)** Your friend suggests the following strategy (this will not be our final strategy). We start with an array R of length b initialized to 0, and a random hash function $h : \mathcal{U} \to \{0, 1, \ldots, b-1\}$. You can assume that h is drawn from some universal hash family, i.e $P(h(x) = h(y)) = 1/b$ for any $x \neq y$. Then the operations are:

   - `update`(x): Increment $R[h(x)]$ by 1.
   - `count`(x): return $R[h(x)]$.

   For every entry $A[i]$ in the list it encounters, the scheme calls `update`$(A[i])$.

   After sequentially processing all n items in the list, what is the expected value of `count`(x)?

   (b) **(2 pt.)** Show that there is a choice of b that is $O(1/\epsilon)$ so that, for any fixed $x \in \mathcal{U}$, we have

   $$\Pr[\texttt{count}(x) < C_x] = 0$$

   and

   $$\Pr[\texttt{count}(x) \geq C_x + \epsilon n] \leq \frac{1}{e}.$$

   [**HINT:** *The first of the requirements is true no matter what b is.*]

   (c) **(2 pt.)** Explain how you would use T copies of the construction in part (a) to define a data structure that, for any fixed $x \in \mathcal{U}$, satisfies (1) with high probability. How big do you need to take T so that the (1) is satisfied with probability at least $1 - \delta$? How much space does your modified construction use? (It should be sublinear in $|\mathcal{U}|$ and n).

Give a complete description and analysis of the data structure, and explain how much space it uses. You may assume that it takes $O(\log |\mathcal{U}|)$ bits to store the hash function h and $O(\log n)$ to store each element in the array R.

(d) Explain how to use your algorithm to solve the following problem:

i. **(4 pt.)** Given a k-sparse vector $a \in \mathbb{Z}_{\geq 0}^N$ ($\mathbb{Z}_{\geq 0}$ is the set of non-negative integers), design a randomized matrix $\Phi \in \mathbb{R}^{m \times N}$ for $m = O(\frac{k \log N}{\epsilon})$ so that the following happens. With probability at least 0.99 over the choice of Φ, you can recover $\tilde{a}$ given Φa, so that simultaneously for all $i \in 1, ..., N$, we have

$$|\tilde{a}[i] - a[i]| \leq \frac{\epsilon \|a\|_1}{2k}.$$

[**HINT:** *Think of the k-sparse vector a as being the histogram of the items in the list A from the previous parts.*]
[**HINT:** *How can you represent a hash function as a matrix multiplication?*]
[**HINT:** *Note that we want a tighter bound, and we want the bound to hold simultaneously for all i. How can we change b and T to achieve this?*]

ii. **(3 pt.)** Now, assuming the above holds for all i, use the k-sparseness of a to construct $\hat{a}$ from $\tilde{a}$ such that

$$\|\hat{a} - a\|_1 \leq \epsilon \|a\|_1.$$

iii. **(0 pt.)** **[This question is zero points, but worth thinking about.]** How does the guarantee in the previous part compare to the RIP matrices (and the compressed sensing guarantee that we can get from them, Theorem 1 in the Lecture 9 lecture notes) that we saw in class? (i.e., is this guarantee weaker? Stronger? Incomparable? The same?)

2. **(6 pt.)[Aquarium apportionment.]**

Suppose that there is a population of n fish, belonging to one of two types: (**A**lgae eaters and **B**arbs). There are also a bunch of fish tanks that these fish could live in. These fish are somewhat picky: each one has a list of at least $1 + \log_2(n)$ fish tanks that it would be happy in (e.g., the algae eater Algernon likes the tank with the mini pirate ship, or the tank with the colorful rocks, but he is not into the tank with the fake ferns). The fishes' lists may be different, and may overlap.

Unfortunately, these two types of fish don't get along well: when an algae eater and a barb are put in the same tank as each other, they will become agitated and fight and stress each other (and you) out.

Use the probabilistic method to show that it is possible to put all of the fish into fish tanks in such a way that each fish is happy with their tank, and none of them fight with each other. That is, define a probability distribution for putting fish into tanks so that, with positive probability, everyone is happy and there is no fighting.

[**HINT:** *If you put each fish in a random tank that they like, the resulting scheme does **not** seem to work...you'll have to come up with a more interesting random distribution.*]

3. **(0 pt.)[Who needs geometry?]**

   This question is worth zero points, but it is an entertainingly obnoxious use of the probabilistic method that you will probably find satisfying.

   Suppose that your friend has – perhaps adversarially – drawn 10 points on the surface of a flat table, which you can think of as an infinite plane. Your friend gives you 10 quarters, which you can think of as discs, all of the same size.

   Your challenge is to place the quarters on the table such that no quarters overlap, *and* every one of your friend's chosen points is covered. Show that this is *always* possible, regardless of the locations of the points.

   One hint is below. Some more hints appear on the **next** page, so that you can avoid spoilers if you wish.

   [**HINT:** *We'll save you some time – trying to think about / classify the possible arrangements of points is not helpful, at least for the purpose of this problem in the context of this course on randomized algorithms. The solution we have in mind behaves the same way regardless of the locations of the points.*]

[**HINT:** *Imagine an arbitrarily large planar sheet of quarters, pressed together as tightly as possible. Such a sheet covers about 90.69% of the plane. See `https://en.wikipedia.org/wiki/Circle_packing` for details.*]

[**HINT:** *The fact that there are 10 points is important, although the significance may not be clear until you start down the right solution path...*]