CS 466/666: Algorithm Design and Analysis **University of Waterloo: Fall 2024**

Lecture 15

October 31, 2024

Instructor: Sepehr Assadi

Topics of this Lecture

1 Compressed Sensing **1**

2 Discrete Sparse Recovery **3**

3 An ℓ_0-Sampler Construction **5**

In the previous lecture, we saw the graph sketching technique for finding a spanning forest in a highly distributed fashion. The key building block of that result were linear sketches for ℓ_0-samplers, which we used without providing their construction. In this lecture, we present a construction of ℓ_0-samplers and prove its correctness. Before getting to them however, it is worth taking a detour and talking about the bigger picture and origins of ℓ_0-samplers, namely, compressed sensing and sparse recovery.

1 Compressed Sensing

Motivation Behind Compressed Sensing

Data analysis is most interesting when the data has structure – there is not much one can do with an unstructured (think of "random") data. But how can we exploit the structure of data? Even before that, what do we mean by "structure" in data?

Structure can mean many things. One major theme in data analysis is *sparsity* – for now, think of sparse data as something that can be compressed much further without losing most of its functionality. A good example is a digital image: one can use standard image compression algorithms (e.g., JPEG) to dramatically reduce the size of the image without changing the visibility of the picture almost at all.

The usual approach to compressing (approximately) sparse data is to first collect the raw data, and then to compress it in software (in the example of an image, the camera first captures an image in the standard pixel basis and only then compress it further via, say, JPEG algorithm). This two-step approach works fine in many applications, but in terms of efficiency, this seems quite counterintuitive: if most of the raw data is going to be thrown out immediately anyway, why did we bother to collect it all in the first place?

The main idea of *compressed sensing* or "compressive sensing" is to *directly* capture data in a compressed form. To put this in the context digital cameras, using compressed sensing we can now use fewer pixels while capturing the photo without degrading image quality, resulting in qualitatively less battery drain for a given number of photos. Another example is in MRI machines. In these machines, the scan time is proportional to the number of measurements (or "slices") taken, and can easily be 30 minutes or more. Compressed sensing techniques have been tried out in some hospitals, and they have sped up scan times by a constant factor.

A Concrete Example: Sparse Recovery

In this lecture, we are going to consider the problem of *sparse recovery* in compressed sensing, which we now define formally. Suppose our target data—often called the *signal*—is a real-valued vector $x \in \mathbb{R}^n$. We have "access" to this data using *linear* measurements. I.e., we can pick a vector $a \in \mathbb{R}^n$ and *observe* $\langle a\,,\, x\rangle$. Our goal is then to recover the signal x using a *minimal* number of measurements. Formally,

Step 1: **Design** m linear measurements $a_1, \ldots, a_m \in \mathbb{R}^m$ or equivalently a matrix $A \in \mathbb{R}^{m\times n}$.

Step 2: An unknown signal $x \in \mathbb{R}^n$ is now picked as input.

Step 3: Receive the measurement results $b = \langle a_1\,,\, x\rangle, \ldots, \langle a_m\,,\, x\rangle$ or equivalently $b := A \cdot x$.

Step 4: **Recover** the unknown signal x from the measurement signal b.

In the setting above, our task as an algorithm designer is to design a suitable *measurement matrix* A and a *recovery algorithm* that allows for recovering x from $A \cdot x$. Note that at this stage, this is an *information theoretic* question not a computational one; in other words, we can, *for now*, ignore the runtime of algorithms for constructing A and recovering x from $A\cdot x$, and solely focus on designing smallest number of measurements possible, i.e., minimize m.

At this level, the problem above has a straightforward solution: set $A = I_n$ to get $b = I_n x = x$ which obviously allows us to recover $x \in \mathbb{R}^n$. This gives us an upper bound of n measurements for recovering x. We also have a lower bound of n measurements: Any system of k equations $A_{k\times n} \cdot x = b$ where $k < n$ is undetermined and thus multiple x satisfy this system, making unique recovery of x from $A \cdot x$ impossible.

However, recall our earlier discussion in compressed sensing. We are typically not interested in recovering arbitrary signals but rather ones with "structure". In particular, in the **sparse recovery** problem, we are guaranteed that the vector x is sparse, namely, has a small number of non-zero entries. Formally, we say that a vector $x \in \mathbb{R}^n$ is k-sparse iff $||x||_0 = k$, i.e., there are only k non-zero entries in x. The k-sparse recovery problem is then defined as follows.

Problem 1 (**Real-Valued Sparse Recovery**)**.** Given parameters $n, k > 0$, *design* a minimal set of measurements $A \in \mathbb{R}^{m\times n}$ such that *for all* k-sparse vectors $x \in \mathbb{R}^n$, x can be uniquely *recovered* from $A \cdot x$.

Note that the lower bound of n on number of measurements no longer hold for this problem because we only need the mapping $x \mapsto A \cdot x$ to be injective for k-sparse vectors and not all vectors in $\mathbb{R}^n$. We are *not* going to consider this problem as it is somewhat beyond the scope of our course and instead we will consider a *discrete* version of it in the next section (which will also have applications to our original ℓ_0-sampler problem). The optimal solution to this problem needs only $\Theta(k \cdot \log(n/k))$ measurements (which are also known to be necessary) and was presented in [CRT06] and [Don06]. We shall also note that these results extend to the more practical setting of approximately recovering an *approximately* k-sparse vector[1].

Remark. In Problem 1, we are designing the measurement matrix A *before* x is even chosen and thus A should work simultaneously for all x. This is often referred to as the **for-all** guarantee in the literature. A "weaker" version of this problem is to design A to recover a *single unknown* x which is chosen independent of A – this weaker guarantee is referred to as the **for-each** guarantee.[a]

[a] This can only make a difference when we use randomization: the for-all guarantee says that the answer is correct simultaneously for all k-sparse x with some fixed probability, while for-each guarantee says that for any one (unknown) signal the answer is correct with fixed probability.

[1] Informally speaking, we say a vector y is k-sparse if $y = x + \sigma$ for a k-sparse vector x and a "noise" signal σ with small ℓ_1 or ℓ_2 (or some other) norms.

2 Discrete Sparse Recovery

We are now going to consider a discrete version of sparse recovery. This is both to gives us some intuition about the real-valued sparse recovery in Problem 1 and also for its applications to the remainder of the course. By discrete sparse recovery, we mean recovery over finite fields and in particular field $\mathbb{F}_2$[2].

Problem 2 (Discrete Sparse Recovery on $\mathbb{F}_2$). Given parameters $n, k > 0$, *design* a minimal set of measurements $A \in \mathbb{F}_2^{m \times n}$ such that *for all* k-sparse vectors $x \in \mathbb{F}_2^n$, x can be uniquely *recovered* from $A \cdot x$.

Before getting to an upper bound for this problem, let us first see what is a natural lower bound on number of measurements. By using m measurements, the set of vectors b that can be given as answer to $A \cdot x$ is 2^m as there are only 2^m m-dimensional vectors in $\mathbb{F}_2^m$. At the same time, we should make sure that for any two different $x \neq y \in \mathbb{F}_2^n$, $A \cdot x \neq A \cdot y$ as otherwise we cannot distinguish x and y from the resulting vector b. Since the number of k-sparse vectors in $\mathbb{F}_2^n$ is $\binom{n}{k}$, we should have,

$$2^m \geqslant \binom{n}{k} \geqslant \left(\frac{n}{k}\right)^k \implies m \geqslant k \cdot \log\left(\frac{n}{k}\right). \qquad \text{(for all } a, b, \tbinom{a}{b} \geqslant \left(\tfrac{a}{b}\right)^b\text{)}$$

As such, $k \cdot \log\left(\frac{n}{k}\right)$ measurements are necessary for this problem.

The natural question at this point is that whether can match this lower bound by an algorithm also, i.e., design $O(k \cdot \log\left(\frac{n}{k}\right))$ measurements that allows us to recover any k-sparse vectors in $\mathbb{F}_2^n$. We will do so in the following.

Warm-Up: $k = 1$ Case

Consider the problem of recovering a *1-sparse* vector $x \in \mathbb{F}_2$. We are going to design $O(\log n)$ measurements for this problem.

For simplicity, we assume n is a power of 2. Consider the set of vectors a_i for $i \in [\log n]$, where:

- a_1 has all 1's in the first $n/2$ coordinates and zero outside.
- a_2 has all 1's in the first $n/4$ and third $n/4$ coordinates and zero outside.
- $\cdots$
- a_i have $n/2^i$ 1's, followed by $n/2^i$ zeros, followed by $n/2^i$ 1's and so on and so forth.

See Figure 1 for an illustration.

This forms the measurement matrix A. We now design the recovery algorithm. Note that A is basically simulating a binary search (suppose j is the index where $x_j = 1$):

- $\langle a_1 , x\rangle = 1$ implies that index j belongs to $[1 : n/2]$ and 0 means it belongs to the rest.
- $\langle a_2 , x\rangle = 1$ implies that index j belongs to $[1 : n/4] \cup [n/2 + 1 : 3n/4]$ and 0 means it belongs to the rest. Combined with the above answer, this identifies an interval of length $n/4$ that j belongs to.
- $\cdots$
- In general, by considering first i rows, we can identify an interval of length $n/2^i$ that contains j.

This way, after $\log n$ rows, we can uniquely recover index j, giving us our recovery algorithm.

[2]Recall that $\mathbb{F}_2$ is the field on $\{0, 1\}$ where computation is mod 2, or equivalently addition is replaced with XOR-operation.

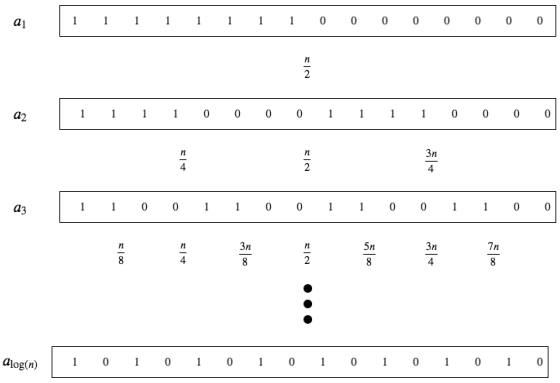

Figure 1: $\log(n)$ vectors in $\mathbb{F}_2^n$ for recovering *1-sparse* vectors

General Case

We are now going to solve the general case of the problem. Extending the binary search approach for $k = 1$ case to general k seems challenging so we will use another approach based on probabilistic arguments.

Theorem 1. *For any $n, k > 0$, there exists a set of $m = O(k \log(\frac{n}{k}))$ measurements $A \in \mathbb{F}_2^{m \times n}$ for recovering any* k-sparse *vector $x \in \mathbb{F}_2^n$.*

This proof is based on the *probabilistic method* which relies on a very basic principle: suppose we sample an object randomly and can show that with *non zero* probability this object satisfies some desired properties; this effectively means that there exists at least one object that satisfies the desired properties, hence proving its existence. In the context of our problem, we are going to pick $A \in \mathbb{F}_2^{m \times n}$ randomly and show that with non-zero probability we can recover x from $A \cdot x$ for all $x \in \mathbb{F}_2^n$. This will prove the existence of a suitable measurement matrix A.

We start with the following claim about taking inner product of a random vector $a \in \mathbb{F}_2^n$ with two *distinct* vectors $x, y \in \mathbb{F}_2^n$. In the following, we use the notation $a \in_R \mathbb{F}_2^n$ to mean that a is chosen uniformly at random from $\mathbb{F}_2^n$.

Claim 2. *For all $x \neq y \in \mathbb{F}_2^n$, and $a \in_R \mathbb{F}_2^n$, $\Pr_a(\langle a, x\rangle = \langle a, y\rangle) = 1/2$.*

Proof. For any index $i \in [n]$ and vector $z \in \mathbb{F}_2^n$, define $z_{-i} = (z_1, z_2, .., z_{i-1}, z_{i+1}, ..z_n)$. Let i be any index where $x_i \neq y_i$. Consider the following process of picking a: first pick $a_{-i} \in \mathbb{F}_2^{n-1}$ and then pick $a_i \in \mathbb{F}_2$.

- Case 1: $\langle a_{-i}, x_{-i}\rangle = \langle a_{-i}, y_{-i}\rangle$: If $a_i = 1$, then $\langle a, x\rangle \neq \langle a, y\rangle$. So, conditioned on this case, with probability $\frac{1}{2}$ we have $\langle a, x\rangle \neq \langle a, y\rangle$.
- Case 2: $\langle a_{-i}, x_{-i}\rangle \neq \langle a_{-i}, y_{-i}\rangle$: If $a_i = 0$, then $\langle a, x\rangle \neq \langle a, y\rangle$. So, conditioned on this case, with probability $\frac{1}{2}$ we have $\langle a, x\rangle \neq \langle a, y\rangle$.

This proves the claim. □

As a corollary of Claim 2, we have that a random matrix can distinguish x and y with "high enough" probability.

Claim 3. *For all $x \neq y \in \mathbb{F}_2^n$, and $A \in_R \mathbb{F}_2^{m \times n}$, $\Pr_A(A \cdot x = A \cdot y) = 1/2^m$.*

Proof. Follows immediately from Claim 2 and independence of the m rows of A. □

Finally, we can use Claim 3 to argue that with non-zero probability, A can distinguish between all pairs of k-sparse vectors $x, y \in \mathbb{F}_2^n$.

Claim 4. *For $m = 2k \log\left(\frac{en}{k}\right)$ and $A \in_R \mathbb{F}_2^{m \times n}$,*

$$\Pr_A \left(\forall k\text{-sparse } x \neq y \in \mathbb{F}_2{}^n \quad Ax \neq Ay\right) > 0.$$

Proof. Since both x,y are k-*sparse* vectors in $\mathbb{F}_2^n$, the number of distinct pairs (x, y) is less than $\binom{n}{k}^2$. Thus, by applying union bound to the events in Claim 3,

$$\Pr_A(\exists x \neq y \in \mathbb{F}_2^n : Ax = Ay) \leqslant \sum_{x \neq y \in \mathbb{F}_2^n} \Pr_A(Ax = Ay) < \frac{\binom{n}{k}^2}{2^m} < \frac{(\frac{en}{k})^{2k}}{2^m} = 1. \qquad \left(\binom{a}{b} \leqslant (\tfrac{ea}{b})^b\right)$$

As the LHS is strictly less than 1, the probability of its complement event is non-zero as desired. □

We can now conclude the proof of Theorem 1.

Proof of Theorem 1. By probabilistic method, Claim 4 implies existence of $A \in \mathbb{F}_2^{m \times n}$ for $m = O(k \log(\frac{n}{k}))$ such that $A \cdot x \neq A \cdot y$ for all k-sparse $x \neq y \in \mathbb{F}_2^n$. We use this matrix as our measurement matrix. For recovery, given $b \in \mathbb{F}_2^m$, we simply go over all $z \in \mathbb{F}_2^n$ and check whether $Az = b$ or not; as the unique answer to this system of equations (among k-sparse vectors) is $Ax = b$, we can recover x uniquely. □

We shall note that in Theorem 1 we only focused on designing minimal number of measurements and in particular ignored the runtime of algorithms. As it is, our algorithms require exponential time to choose the measurement matrix A, and more importantly, also take exponential time to do the recovery. We revisit this issue in problem set 2.

Remark. Note that the proof of Theorem 1 used k-sparsity in a very simple way by simply bounding number of vectors that we aim to do the recovery for by $\binom{n}{k}$. It is thus easy to see that we can extend this theorem for performing recovery of any subset $\mathcal{X} \subseteq \mathbb{F}_2^n$ using $O(\log |\mathcal{X}|)$ measurements.

3 An ℓ_0-Sampler Construction

With the above context in mind, we are now ready to get back to our original ℓ_0-sampler problem from the previous lecture. In particular, we will prove the following theorem used in the design of our algorithm in Lecture 14.

Theorem 5. *For every $M \geqslant 1$ and sufficiently large $m \in \mathbb{N}$, there exists a distribution D such that the following is true. For* each *vector $x \in \{-M, \ldots, 0, \ldots, M\}^m$, if we sample A from D* independent *of x, then, with high probability, using only $A \cdot x$ and A, we can recover a pair (i, x_i) such that i is chosen uniformly at random from the support of x and $x_i \neq 0$ is its value. Moreover, $A \cdot x$ has size $O(\log^4(mM))$ bits.*

We will prove the theorem in a couple of steps.

Step 1: When $||x||_0 = 1$. In this case, there is a very simple solution. Let

$$a_1 := [1, 1, 1, \ldots, 1]$$
$$a_2 := [1, 2, 3, \ldots, m]$$
$$A_1 := \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}.$$

Then, if the only non-zero entry of x is x_i, we will get,

$$A_1 \cdot x = \begin{bmatrix} \langle a_1 \, , \, x \rangle \\ \\ \langle a_2 \, , \, x \rangle \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{m} 1 \cdot x_j \\ \\ \sum_{j=1}^{m} j \cdot x_j \end{bmatrix} = \begin{bmatrix} x_i \\ \\ i \cdot x_i \end{bmatrix}.$$

Thus, if we denote $A_1 \cdot x = [b_1; b_2]$, we get that b_1 is the value of x_i and b_2/b_1 is the index of i. I.e., the output should be $(b_2/b_1, b_1) = (i, x_i)$.

In this case, the matrix A_1 has two rows and the vector $A_1 \cdot x$ has size $O(\log(mM))$ bits.

Step 2: How to test if $||x||_0 = 1$? Step 1 implies that if $||x||_0 = 1$, then we can solve the problem quite easily. But, can we even check if we are in this "good" case? We will do so using a simple randomized strategy.

Sample a $3 \times m$ dimensional matrix A where every column has exactly one 1 and two 0's and the position of the 1 is chosen independently and uniformly at random. Consider $A \cdot x = [b_1; b_2; b_3]$.

Suppose first that $||x||_0 = 1$. Then, exactly one of b_1 or b_2 or b_3 is non-zero. In particular, if i is the index of the non-zero entry of x, then, the row of A on column i that has the value 1 corresponds to the non-zero entry in $A \cdot x$ also.

Now, suppose that $||x||_0 \geqslant 2$. Let $i \neq j$ be two non-zero indices from x. Define the vector $y(i)$ where $y(i)_i = x_i$ and $y(i)$ is zero everywhere else. Define $y(j)$ similarly with $y(j)_j = x_j$ and zero everywhere else. And define $y = x - y(i) - y(j)$. Fix all columns of A other that i-th and j-th column, which also fixes the value of $A \cdot y = [c_1; c_2; c_3]$. We now have,

$$A \cdot x = A \cdot y + A \cdot y(i) + A \cdot y(j) = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} + \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} + \begin{bmatrix} j_1 \\ j_2 \\ j_3 \end{bmatrix},$$

where exactly one of i_1, i_2, i_3 (resp, j_1, j_2, j_3) is non-zero depending which row of A in the column i (resp. the column j) receives the value of 1. Note that these choices are still independent and uniform at random for both i and j.

Claim 6. *In this case, with probability at least $1/9$, at least two values in b_1, b_2, b_3 are non-zero.*

Proof. We say the event 'success' has happened if at least two-values in b_1, b_2, b_3 are non-zero. We consider different cases based on c_1, c_2, c_3.

- All of c_1, c_2, c_3 are equal to zero: In this case, if 1 of column i is different than 1 of column j, then, two different values in b_1, b_2, b_3 are going to end up non-zero. Thus, the probability of success is:

$$\Pr(\text{success}) \geqslant \Pr(\text{1 of column } i \text{ is different from 1 of column } j) = \frac{2}{3} > \frac{1}{9}.$$

- Two of c_1, c_2, c_3 are equal to zero: by symmetry, let us assume $c_1 = c_2 = 0$ and $c_3 \neq 0$. If 1 column i is different from 1 of column j and both are different from 3, then, all three of b_1, b_2, b_3 will be non-zero and success happens. Thus,

$$\Pr(\text{success}) \geqslant \Pr(\text{1 of column } i \text{ 1 and 1 of column } j \text{ is 2 or vice versa}) = 2 \cdot \frac{1}{9} > \frac{1}{9}.$$

- One of c_1, c_2, c_3 is equal to zero: by symmetry, let us assume $c_1 = 0$ and $c_2, c_3 \neq 0$. If 1 of column i and 1 of column j both go to index 1, then, both of b_2, b_3 remain non-zero also. Thus,

$$\Pr(\text{success}) \geqslant \Pr(\text{1 of column } i \text{ 1 and 1 of column } j \text{ both go to index 1}) = \frac{1}{9}.$$

- Finally, none of c_1, c_2, c_3 are zero. In this case, as long as 1 of column i and 1 of column j go to the same index, then both of the remaining indices remain non-zero in b_1, b_2, b_3. Thus,

$$\Pr(\text{success}) \geqslant \Pr(1 \text{ of column } i \text{ 1 and 1 of column } j \text{ go to the same index}) = \frac{1}{3} > \frac{1}{9}.$$

This concludes the proof. □

Finally, in the case that $||x||_0 = 0$, we always have $A \cdot x = 0$.

Putting these results together, we have the following *distinguisher*: if $[b_1, b_2, b_3]$ has exactly one non-zero entry, we consider it a $||x||_0 = 1$ case, and otherwise, we consider the input not having this property. Now, suppose we pick a matrix A_2 which consists of $t = 90 \ln m$ independent copies of the matrix A. When $||x||_0 = 1$, *all* the copies return 'one non-zero entry' case, while when $||x||_0 = 1$, the probability of this even happening is, by Claim 6, at most

$$\left(1 - \frac{1}{9}\right)^{90 \ln m} \leqslant \exp(-10 \ln m) = m^{-10}.$$

So, with high probability, we can solve the problem correctly in this case.

In this case, the matrix A_2 has $270 \ln m$ rows and the vector $A_2 \cdot x$ has size $O(\log^2(mM))$ bits.

Step 3: when $2^k \leqslant ||x||_0 \leqslant 2^{k+1}$ for some given $k \geqslant 0$. We will attempt to reduce this case to the case of step 1 while also running the step 2 *test* in parallel to make sure the reduction indeed worked correctly. First pick a *diagonal* matrix B with dimensions $m \times m$ where each entry B_{ii} for $i \in [m]$ is 1 with probability $1/2^{k+1}$ and 0 otherwise. Let $y := B \cdot x$.

Claim 7. *With probability at least* $1/8$, $||y||_0 = 1$.

Proof. Let $s = ||x||_0$. For $||y||_0 = 1$, we need *exactly* one of the indices i in the support of x to be sampled on the diagonal of B. Thus,

$$\Pr(||y||_0 = 1) = s \cdot \frac{1}{2^{k+1}} \cdot \left(1 - \frac{1}{2^{k+1}}\right)^{s-1} \geqslant \frac{1}{2} \cdot \left(1 - \frac{1}{2^{k+1}}\right)^{2^{k+1}} > \frac{1}{2 \cdot e} > \frac{1}{8}.$$

□

Now, use the matrices A_1 and A_2 from the previous step to create the matrix

$$A := \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \cdot B.$$

We have

$$A \cdot x = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \cdot B \cdot x = \begin{bmatrix} A_1 \cdot y \\ A_2 \cdot y \end{bmatrix}.$$

Thus, *if* y has exactly one non-zero entry, then, we can recover that non-zero entry from $A_1 \cdot y$ and $A_2 \cdot y$ allows us to test with high probability that $||y||_0 = 1$ or not.

The final matrix we create in this step is then consists of copying A $80 \ln m$ times independently on top of each other to get a matrix A_3. The probability that A_3 cannot recover an index because non of the y-vectors it creates have $||y||_0 = 1$ is at most (by Claim 7):

$$\left(1 - \frac{1}{8}\right)^{80 \ln m} \leqslant \exp(-10 \ln m) = m^{-10}.$$

Moreover, since there are only $O(\log m)$ 'tests' for $||y||_0 = 1$, with high probability, all of them succeeds and thus the algorithm does not return a wrong answer.

In this case, the matrix A_3 has $O(\log^2 m)$ rows and the vector $A_3 \cdot x$ has size $O(\log^3(mM))$ bits.

Step 4: arbitrary x. We can now solve the general problem. While we do not know the value of k, there are only $\log m$ different choices for it for choices of k in $1 \leqslant 2^k \leqslant m$. So, we simply create matrices A'_k for $k = 1$ to $\log m$, where each A'_k is the matrix we created in step 3 for the value of k. We then set A_4 to be these matrices stacked on top of each other.

For the "correct" choice of k, the matrix A'_k, by the argument in step 3 returns a correct solution with high probability. For all the other indices, we never run the matrix A_1 and its recovery phase before getting ensured by matrix A_2 that the number of non-zero entries passed to A_1 is only 1, hence, with high probability, we will never return a wrong answer.

Finally, the matrix A_4 has $O(\log^3 m)$ rows and the vector $A_3 \cdot x$ has size $O(\log^4 (mM))$ bits.

This concludes the proof of Theorem 5. Before finishing this lecture, we should mention that the bounds in Theorem 5 are not optimal and one can construct even more efficient ℓ_0-samplers [JST11].

References

[CRT06] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Information Theory*, 52(2):489–509, 2006. 2

[Don06] David L. Donoho. Compressed sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, 2006. 2

[JST11] Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 49–58. ACM, 2011. 8