Lecturers: Hung Q. Ngo and Atri Rudra
Last update: October 13, 2011

A recursive construction of efficiently decodable list-disjunct matrices

The method described here is from Ngo-Porat-Rudra [1].

1 Main idea

Let $\mathbf{M}$ be a $t \times N$ (d, ℓ)-list-disjunct matrix. Given a test outcome vector $\mathbf{y} \in \{0,1\}^t$, the naive decoder can identify the set $R_{\mathbf{y}}$ of items which contains all the positives plus $< \ell$ negative items in time $O(tN)$. From the probabilistic proof in the previous lecture, we also know that we can achieve $t = O(d^2/\ell \log(N/(d+\ell)))$. For small d and ℓ, most of the running time of the naive decoding algorithm is "spent" on the linear term N. How do we reduce the running time down to sub-linear? Say even as high as $\tilde{O}(\text{poly}(d) \cdot \sqrt{N})$?

For concreteness, let us consider a special case when $d = \ell$. For every $d \leq N/2$, there exists a (d,d)-list-disjunct matrix with $t = O(d \log(N/d))$ rows which is decodable in time $O(Nd \log(N/d))$. We shall use this knowledge to construct a (d,d)-list-disjunct matrix which is decodable in sub-linear time as follows.

Imagine putting all N blood samples on a $\sqrt{N} \times \sqrt{N}$ grid. Let us index the samples by pairs $[r,c]$ for integers $1 \leq c, r \leq \sqrt{N}$. Every row and every column contains exactly $\sqrt{N}$ blood samples each. For each column c, create a "column sample" labeled $[*,c]$ by pooling together *all* samples $[r,c]$ for $r \in [\sqrt{N}]$. We thus have $\sqrt{N}$ column samples $[*,c]$, each of which is a set of $\sqrt{N}$ individual samples. Similarly, we have $\sqrt{N}$ "row samples" labeled $[r,*]$ for $r \in [\sqrt{N}]$. A column/row sample is said to be positive if it contains at least one positive sample.

Now, let $\mathbf{M}_h$ (h is for *horizontal*) and $\mathbf{M}_v$ (v) is for *vertical*) be two $t_1 \times \sqrt{N}$ (d,d)-list-disjunct matrices. Let $\mathbf{M}^*$ be any $t_2 \times N$ (d,d)-list-disjunct matrix. We use $\mathbf{M}_v$ to test the column samples, use $\mathbf{M}_h$ to test the row samples, and use $\mathbf{M}^*$ to test the normal samples. In total, we have $t = 2t_1 + t_2$ (perhaps adaptive) tests.

After running the naive decoding algorithm on $\mathbf{M}_v$ and $\mathbf{M}_h$ in time $O(t_1\sqrt{N})$, we can identify a set C of less than $2d$ column samples $[*,c]$ which contain all the positive column samples; and, we can identify a set R of less than $2d$ row samples $[r,*]$ which contain all the positive row samples. The positive individual samples *must* lie in the intersection of the columns C and the rows R. There are totally $2d \times 2d$ candidate samples in the intersections. Thus, we have narrowed down N to a set of at most $4d^2$ samples.

Finally, we run the naive decoding algorithm on $\mathbf{M}^*$ restricted only to these $4d^2$ samples to get a set of at most $2d$ candidates. The running time of the decoding algorithm on $\mathbf{M}^*$ is only $t \times (4d^2)$. In total, the running time is $O(t_1\sqrt{N} + td^2)$. Since we can achieve $t_1 = O(d \log(\sqrt{N}/d))$ and $t_2 = O(d \log(N/d))$, the total decoding time is about $O((d\sqrt{N} + d^3)\log(N/d))$, which is sublinear in N for small d.

2 Turning the idea into a non-adaptive system

We want a non-adaptive group testing strategy. The idea described above does not work as is. What we need is a single $t \times N$ matrix which is (d,d)-list-disjunct, not a few matrices of different dimensions. How do we

realize the idea of testing column samples and row samples? It turns out we only need the *structures* of the matrices $\mathbf{M}_v, \mathbf{M}_h$, but we do not need to use them directly.

From the matrix $\mathbf{M}_v$ – which again is of dimension $t_1 \times \sqrt{N}$ – construct a matrix $\mathbf{M}_v^*$ of dimension $t_1 \times N$ as follows. Each row i of $\mathbf{M}_v^*$ represents a test (pool) of $\mathbf{M}_v$. If the test i in $\mathbf{M}_v$ contains a column sample $[*, c]$ then we put *all* items labeled $[r, c]$ in the ith test of $\mathbf{M}_v^*$. In other words, there is a 1 in row i and column $[r, c]$ of $\mathbf{M}_v^*$ if and only if there is a 1 in row i and column $[*, c]$ of $\mathbf{M}_v$. It is not hard to see that a test using $\mathbf{M}_v^*$ turns positive if and only if the corresponding test using $\mathbf{M}_v$ (on the column samples) turns positive. Similarly, we construct the matrix $\mathbf{M}_h^*$ of dimension $t_1 \times N$. Our final group testing matrix $\mathbf{M}$ is constructed by stacking vertically three matrices $\mathbf{M}_v^*, \mathbf{M}_h^*$, and $\mathbf{M}^*$, for a total of $t = 2t_1 + t_2$ rows.

The decoding strategy is to use the rest results corresponding to the rows of $\mathbf{M}_h^*$ and $\mathbf{M}_v^*$ to identify a set of at msot $4d^2$ candidate columns of $\mathbf{M}$, and then run the naive decoder on the $\mathbf{M}^*$ part restricted to these $4d^2$ columns. The total decoding time is $O(t_1\sqrt{N} + d^2 t_2)$. Formally, the above reasoning leads to the following lemma.

Lemma 2.1. *Let d be a positive integer. Suppose for $N \geq d^2$, there exists a (d, d)-list-disjunct matrix $\mathbf{M}_{\sqrt{N}}$ of size $t(d, \sqrt{N}) \times \sqrt{N}$ which can be decoded in time $D(d, \sqrt{N})$; and, there exists a (d, d)-list-disjunct matrix $\mathbf{M}^*$ of size $t^*(d, N) \times N$. Then, there exists a (d, d)-list-disjunct matrix $\mathbf{M}_N$ with*

$$t(d, N) = 2t(d, \sqrt{N}) + t^*(d, N)$$

rows and N columns which can be decoded in time

$$D(d, N) = 2D(d, \sqrt{N}) + 4d^2 \cdot t^*(d, N).$$

Next, we apply the above lemma recursively. Let us apply recursion a few times to see the pattern. First, from the probabilistic method proof we discussed in a previous lecture, we know there exists an absolute constant c such that, for every $d \leq N$, there is a (d, d)-list-disjunct matrix $\mathbf{M}^*(d, N)$ of dimension $t^*(d, N) \times N$ where $t^*(d, N) = cd \log_2(N/d)$.

- When $N = d^2$, let $\mathbf{M}_{\sqrt{N}} = \mathbf{M}_d$ be the $d \times d$ identity matrix which is certainly (d, d)-list-disjunct with $t(d, d) = d$ rows and which can be decoded in time $D(d, d) = d$. Now, using $\mathbf{M}^*(d, d)$ and the lemma, we obtain a matrix $\mathbf{M}_{d^2}$ with dimension $t(d, d^2) \times d^2$ and decoding time $D(d, d^2)$, where

$$\begin{aligned} t(d, d^2) &= 2d + cd \log d \\ D(d, d^2) &= 2d + (4d^2) \cdot cd \log d \end{aligned}$$

- When $N = d^4$, applying the lemma using $\mathbf{M}_{\sqrt{N}} = \mathbf{M}_{d^2}$ just constructed above, we obtain

$$\begin{aligned} t(d, d^4) &= 4d + 2cd \log(d) + cd \log(d^3) \\ D(d, d^4) &= 4d + 8cd^3 \log d + (4d^2) \cdot cd \log(d^3) \end{aligned}$$

- Generally, for $N = d^{2^i}$ we have the following recurrences

$$\begin{aligned} t(d, d^{2^i}) &= 2t(d, d^{2^{i-1}}) + cd \log(d^{2^i - 1}) \\ D(d, d^{2^i}) &= 2D(d, d^{2^{i-1}}) + 4cd^3 \log(d^{2^i - 1}). \end{aligned}$$

  The base case is when $i = 0$, where we have $t(d, d) = d$ and $D(d, d) = d$.

When $N = d^{2^i}$, note that $2^i = \log_d N$ and $i = \log\log_d N$. To solve for $t(d, d^{2^i})$, we iterate the recursion as follows.

$$\begin{aligned}
t(d, d^{2^i}) &= 2t(d, d^{2^{i-1}}) + cd\log(d^{2^i-1}) \\
&= 2t(d, d^{2^{i-1}}) + (2^i - 1)cd\log d \\
&= 2\left[2t(d, d^{2^{i-2}}) + (2^{i-1} - 1)cd\log d\right] + (2^i - 1)cd\log d \\
&= 4t(d, d^{2^{i-2}}) + [2\cdot 2^i - (1+2)]cd\log d \\
&= 4\left[2t(d, d^{2^{i-3}}) + (2^{i-2} - 1)cd\log d\right] + [2\cdot 2^i - (1+2)]cd\log d \\
&= 8t(d, d^{2^{i-3}}) + [3\cdot 2^i - (1+2+4)]cd\log d \\
&= \cdots \\
&= 2^i t(d,d) + [i2^i - (1+2+4+\cdots+2^{i-1})]cd\log d \\
&= 2^i t(d,d) + [i2^i - 2^i + 1]cd\log d \\
&\leq= 2^i t(d,d) + [i2^i - i]cd\log d \\
&= d\log_d N + (\log\log_d N)\cdot cd\log(N/d).
\end{aligned}$$

For $d \geq 2$, obviously $\log_d N = O(\log(N/d)$. Hence,

$$t(d, d^{2^i}) = O\left((\log\log_d N)\cdot d\log(N/d)\right).$$

What about the case when $N \neq d^{2^i}$ for some integer i? For $N \geq d^2$, let $i = \lceil \log\log_d N \rceil$, then $d^{2^{i-1}} < N \leq d^{2^i}$. This means $i < \log\log_d N + 1$. We construct $\mathbf{M}_N$ by removing from $\mathbf{M}_{d^{2^i}}$ arbitrarily $d^{2^i} - N$ columns. The resulting matrix can be operated on as if those removed columns were all negative items. It is not hard to see that

$$t(d, N) = O\left((\log\log_d N)\cdot d\log(N/d)\right).$$

What is really nice about this result is that we now have an efficiently decodable (d,d)-list-disjunct matrix $\mathbf{M}_N$ for every $N \geq d^2$ whose number of rows is only brown up from the optimal $O(d\log(N/d))$ by a doulbe-log factor of $\log\log_d N$.

Exercise 1. Using the above recurrence, show that

$$D(t, N) = O\left((\log\log_d N)d^3\log(N/d)\right).$$

The above analysis leads to the following result.

Theorem 2.2. *For any positive integers $d < N$, there exists a (d,d)-list-disjunct matrix $\mathbf{M}$ with*

$$t = O\left((\log\log_d N)\cdot d\log(N/d)\right)$$

rows and N columns. Furthermore, $\mathbf{M}$ can be decoded in time

$$O\left((\log\log_d N)\cdot d^3\log(N/d)\right).$$

3 Generalizing the main idea

The above ideas can be greatly generalized. The following section can be skipped without losing much of the intuition behind the results.

Theorem 3.1. *Let $n \geq d \geq 1$ be integers. Assume for every $i \geq d$, there is a (d, ℓ)-list disjunct $t(i) \times i$ matrix $\mathbf{M}_i$ for integers $1 \leq \ell \leq n - d$. Let $1 \leq a \leq \log n$ and $1 \leq b \leq \log n/a$ be integers. Then there exists a $t_{a,b} \times n$ matrix $\mathbf{M}_{a,b}$ that is (d, ℓ)-list disjunct that can be decoded in time $D_{a,b}$ where*

$$t_{a,b} = \sum_{j=0}^{\lceil \log_b(\frac{\log n}{a}) \rceil - 1} b^j \cdot t\left(\sqrt[b^j]{n}\right) \tag{1}$$

and

$$D_{a,b} = O\left(t_{a,b} \cdot \left(\frac{\log n \cdot 2^a}{a} + (d+\ell)^b\right)\right). \tag{2}$$

Finally, if the family of matrices $\{\mathbf{M}_i\}_{i \geq d}$ is (strongly) explicit then so is $\mathbf{M}_{a,b}$.

Proof. We will construct the final matrix $\mathbf{M}_{a,b}$ recursively. In particular, let such a matrix in the recursion with N columns be denoted by $\mathbf{M}_{a,b}(N)$. Note that the final matrix is $\mathbf{M}_{a,b} = \mathbf{M}_{a,b}(n)$. (For notational convenience, we will define $D_{a,b}(N)$ and $t_{a,b}(N)$ to be the decoding time for and the number of rows in $\mathbf{M}_{a,b}(N)$ respectively). Next, we define the recursion.

If $N \leq 2^a$, then set $\mathbf{M}_{a,b}(N) = \mathbf{M}_N$. Note that in this case, $t_{a,b}(N) = t(N)$. Further, we will use the naive decoder in the base case, which implies that $D_{a,b}(N) = O(t_{a,b}(N) \cdot N) \leq O(2^a \cdot t_{a,b}(N))$. It is easy to check that both (1) and (2) are satisfied. Finally because $\mathbf{M}_N$ is a (d, ℓ)-list disjunct matrix, so is $\mathbf{M}_{a,b}(N)$.

Now consider the case when $N > 2^a$. For $i \in [b]$, define $\mathbf{M}^{(i)}$ to be the $t_{a,b}(\sqrt[b]{N}) \times N$ matrix whose kth column (for $k \in [N]$) is identical to the mth column in $\mathbf{M}_{a,b}(\sqrt[b]{N})$ where m is the ith chunk of $\frac{1}{b}\log n$ bits in k (again, we think of k and m as their respective binary representations). Define $\mathbf{M}_{a,b}(N)$ to be the stacking of $\mathbf{M}^{(1)}, \mathbf{M}^{(2)}, \ldots, \mathbf{M}^{(b)}$ and $\mathbf{M}_N$. Since $\mathbf{M}_N$ is a (d, ℓ)-list disjunct matrix, so is $\mathbf{M}_{a,b}(N)$. Next, we verify that (1) holds. To this end note that

$$t_{a,b}(N) = b \cdot t_{a,b}(\sqrt[b]{N}) + t(N). \tag{3}$$

In particular, (by induction) all the $\mathbf{M}^{(i)}$ contribute

$$b \cdot \sum_{j=0}^{\left\lceil \log_b\left(\frac{\log \sqrt[b]{N}}{a}\right)\right\rceil - 1} b^j \cdot t\left(\sqrt[b^j]{\sqrt[b]{N}}\right) = \sum_{j=1}^{\lceil \log_b(\frac{\log N}{a}) \rceil - 1} b^j \cdot t\left(\sqrt[b^j]{N}\right)$$

rows. Since $\mathbf{M}_N$ adds another $t(N)$ rows, $\mathbf{M}_{a,b}(N)$ indeed satisfies (1). Finally, we consider the decoding of $\mathbf{M}_{a,b}(N)$. The decoding algorithm is natural: we run the decoding algorithm for $\mathbf{M}_{a,b}(\sqrt[b]{N})$ (that is guaranteed by induction) on the part of the (erroneous) outcome vector corresponding to each of the $\mathbf{M}^{(i)}$ ($i \in [b]$) to compute sets S_i with the following guarantee: each of the at most d defective indices $k \in [N]$ projected to the ith chunk of $\frac{1}{b}\log n$ is contained in S_i. Finally, we run the naive decoding algorithm for $\mathbf{M}_N$ on $\mathcal{S} \stackrel{def}{=} S_1 \times S_2 \times \cdot \times S_b$ (note that by definition of S_i all of the defective items will be in $\mathcal{S}$). To complete the proof, we need to verify that this algorithm takes time as claimed in (2). Note that

$$D_{a,b}(N) = b \cdot D_{a,b}(\sqrt[b]{N}) + O\left(|\mathcal{S}| \cdot t(N)\right).$$

By induction, we have

$$D_{a,b}(\sqrt[b]{N}) = O\left(t_{a,b}(\sqrt[b]{N}) \cdot \left(\frac{\log N \cdot 2^a}{a} + (d+\ell)^b\right)\right),$$

and since $\mathbf{M}_{a,b}(\sqrt[b]{N})$ is a (d,ℓ)-list disjunct matrix, we have

$$|\mathcal{S}| \leq (d+\ell)^b.$$

The three relations above along with (3) show that $D_{a,b}(N)$ satisfies (2), as desired.

Finally, the claim on explicitness follows from the construction. □

The bound in (1) is somewhat unwieldy. We note in Corollary 3.2 that when $t(i) = d^x \log^y i$ for some reals $x, y \geq 1$, we can achieve efficient decoding with only a log-log factor increase in number of tests. We will primarily use this result in our applications.

Note that in this case the bound in (1) can be bounded as

$$\sum_{j=0}^{\lceil \log_b(\frac{\log n}{a})\rceil - 1} b^j \cdot d^x \cdot \left(\frac{\log n}{b^j}\right)^y \leq \left\lceil \log_b\left(\frac{\log n}{a}\right)\right\rceil \cdot d^x \log^y n.$$

Theorem 3.1 with $b = 2$ and $a = \log d$, along with the observation above, implies the following:

Corollary 3.2. *Let $n \geq d \geq 1$ be integers and $x, y \geq 1$ be reals. Assume for every $i \geq d$, there is a (d,ℓ)-list disjunct $O(d^x \log^y i) \times i$ matrix for integers $1 \leq \ell \leq n-d$. Then there exists a $t \times n$ matrix that is (d,ℓ)-list disjunct that can be decoded in $\mathrm{poly}(t,\ell)$ time, where*

$$t \leq O\left(d^x \cdot \log^y n \cdot \log\log_d n\right).$$

Finally, if the original matrices are (strongly) explicit then so is the new one.

In other words, the above result implies that we can achieve efficient decoding with only a log-log factor increase in number of tests. We will primarily use the above result in our applications.

To show the versatility of Theorem 3.1 we present another instantiation. For any $0 \leq \epsilon \leq 1$, if we pick $a = b\log(d+\ell)$ and $b = (\log n/a)^\epsilon$, we get the following result:

Corollary 3.3. *Let $n \geq d \geq 1$ be integers and $x, y \geq 1$, $0 < \epsilon \leq 1$ be reals. Assume for every $i \geq d$, there is a (d,ℓ)-list disjunct $O(d^x \log^y i) \times i$ matrix for integers $1 \leq \ell \leq n-d$. Then there exists a $t \times n$ matrix that is (d,ℓ)-list disjunct that can be decoded in*

$$\mathrm{poly}(t,\ell) \cdot 2^{\sqrt[1+\epsilon]{\log^\epsilon n \cdot \log(d+\ell)}}$$

time, where

$$t \leq O\left(\frac{1}{\epsilon} \cdot d^x \log^y n\right).$$

Finally, if the original matrices are (strongly) explicit then so is the new one.

Note that the above result implies that with only a constant factor blow-up in the number of tests, one can perform sub-linear in n time decoding when $(d+\ell)$ is polynomially small in n.

References

[1] H. Q. Ngo, E. Porat, and A. Rudra, *Efficiently decodable error-correcting list disjunct matrices and applications - (extended abstract)*, in ICALP (1), 2011, pp. 557–568.