Lecturers: Hung Q. Ngo and Atri Rudra
Last update: October 11, 2011

Efficient Decoding and List Group Testing

1 Towards efficiently decodable group testing

History. Recall the four objectives of group testing: (1) minimize the number of tests, (2) (Strongly) explicit construction of the group testing matrix, (3) fast decoding, and (4) error-correcting.

With respect to the number of tests, we have shown that $\Omega\left(\frac{d^2 \log N}{\log d}\right) = t(d, N) = O\left(d^2 \log N\right)$, where $t(d, N)$ is the minimum number of rows of a $t \times N$ d-disjunct matrix. Hence, we will strive to construct matrices with t as close to $O(d^2 \log N)$ as possible.

With respect to explicit constructions of group testing matrices, a randomly chosen matrix is d-disjunct with $t = O\left(d^2 \log N\right)$ rows with high probability. However, that fact does not tell us *how* to obtain such a matrix. We have made two connections from d-disjunct matrix construction to the k-RESTRICTION problem and the SET-COVER problem. Both of these connections allow us to construct d-disjunct matrices with $t = O(d^2 \log N)$ rows; however, the running time is $\Omega(N^d)$ which is not so great. By derandomizing the probabilistic proof of the Gilbert-Varshamov bound, we can construct a d-disjunct matrix in time $\tilde{O}(tN)$ with $t = O(d^2 \log N)$. This result by [14] is almost the best one can hope for. An obvious open problem is to design a strongly explicit construction of d-disjunct matrices attaining $t = O(d^2 \log N)$. This problem seems to be within reach of the current "technologies" we have.

In the next few lectures, we will look at the problem of constructing d-disjunct matrices which can be decoded in time sub-linear in N. Recall that a the *naive decoding* algorithm on a d-disjunct matrix can identify all the positives in $O(tN)$ time. However, this decoding time is too long for some applications. We will develop tools to describe two methods for constructing group testing matrices which can be decoded efficiently (in time $\text{poly}(d, \log N)$). The first method, also from [13], is based on a simple recursive construction. The second method, based on code concatenation, was first observed in Indyk-Ngo-Rudra [11] and exploited to its full potential in Ngo-Porat-Rudra [13]. The common ingredient of both methods is a combinatorial object called *list-disjunct matrix*. There is also a notion of *list-separable matrix* which is closely related to list-disjunct matrix. List-disjunct and list-separable matrices are interesting on their own, with applications beyond the group testing problem. We shall discuss the applications in a later lecture. The other key ingredient of the second method is the notion of *list-recoverable codes*, which will be introduced in later lectures too.

Main idea. To construct an efficiently decodable group testing matrix, the main idea is to stack on top of one another a "filtering" matrix $\mathbf{F}$ and an "identification" matrix $\mathbf{D}$. (See Figure 1.) The filtering matrix is used to identify quickly a "small" set of L candidate items which include *all* the positives. Then, the identification matrix is used to pinpoint precisely the positives. For example, let $\mathbf{D}$ be any d-disjunct matrix and $\mathbf{F}$ be any matrix satisfying the filtering property above. Further assume that from the tests corresponding to the rows of $\mathbf{F}$ we can produce a set S of $L = \text{poly}(d, \log N)$ candidate items in time $\text{poly}(d, \log N)$. Then, by running the naive decoding algorithm on S using $\mathbf{D}$ and the test results corresponding to $\mathbf{D}$, we can identify all the positives in time $\text{poly}(d, \log N)$. This idea is analogous to the constructions of (efficiently

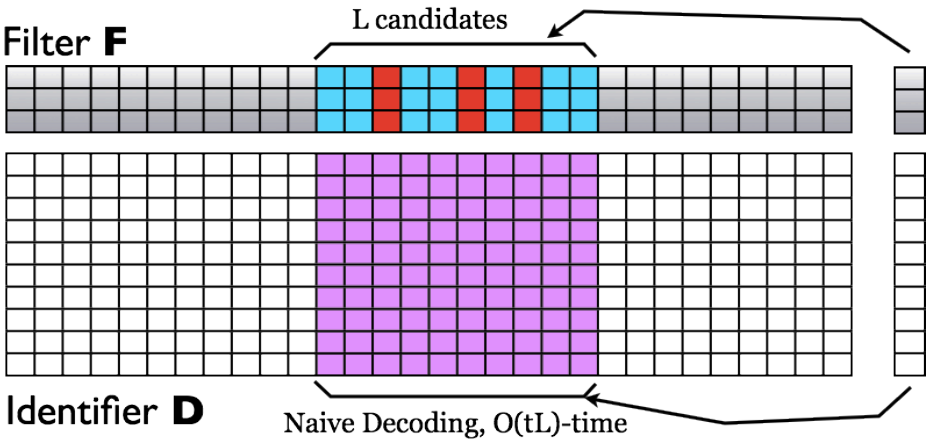

Figure 1: The main idea behind fast decoding algorithms.

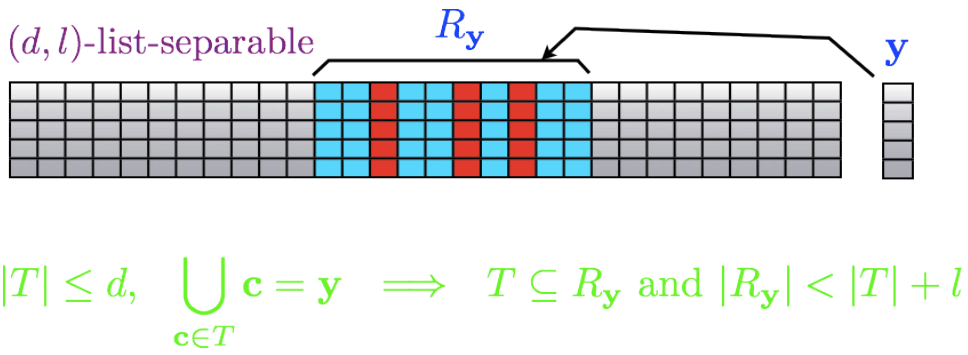

Figure 2: Illustration of the list-separable matrix definition.

decodable or not) compressive sensing schemes [3, 6, 9] which we will cover next semester in the second part of this seminar.

2 List Separable and List Disjunct Matrices

The first problem is how to formalize the notion of "filtering matrix." In coding theory, producing a small list of candidate codewords is the *list decoding problem* [17]. We borrow the intuition from list decoding and from separable matrices to define the following notion "filtering matrices".

Definition 2.1 ((d,ℓ)-list-separable)**.** Given positive integers t, d, ℓ, N where $d + l \leq N$, a $t \times N$ binary matrix $\mathbf{M}$ is said to be (d,l)-*list-separable* if it satisfies the following property. For any $\mathbf{y} \in \{0,1\}^t$ there exists a column set $R_{\mathbf{y}}$ such that, if T is any set of at most d columns of $\mathbf{M}$ whose union is $\mathbf{y}$, then $T \subseteq R_{\mathbf{y}}$ and $|R_{\mathbf{y}}| < |T| + \ell$. Note that a $(d,1)$-list-separable matrix is precisely a d-separable matrix! Figure 2 illustrates the definition.

Suppose we have a group testing matrix with a corresponding decoding algorithm which produces for every outcome vector $\mathbf{y} \in \{0,1\}^t$ a candidate set $R_{\mathbf{y}}$ including *all* the positives plus $< \ell$ negative items,

then the matrix is (d, l)-list- separable. (If $\mathbf{y}$ does not correspond to any outcome vector then we can set $R_{\mathbf{y}} = \emptyset$.) The next natural line of development is to define and study list-disjunct matrices, because from the definition of list-separable matrices it is not clear how we can obtain the "list" $R_{\mathbf{y}}$ given an outcome vector $\mathbf{y}$.

Definition 2.2 ((d, ℓ)-list-disjunct)**.** A matrix $\mathbf{M}$ is called (d, l)-list-disjunct if the naive decoding algorithm always returns an item set which includes *all* the (at most d) positives plus $< \ell$ negative items. Recall that the naive decoding algorithm says: eliminate all items which participate in a negative test, and return the remaining items.

So (d, ℓ)-list-disjunct matrices admit $O(tN)$-decoding time. However, we will need a more precise characterization of list-disjunct matrices to actually construct them *and* construct ones which allow for sub-linear decoding time.

Proposition 2.3. *Let $d + \ell \leq N$ be positive integers. A matrix $\mathbf{M}$ is (d, ℓ)-list-disjunct if and only if, for any two disjoint sets S and T of columns of $\mathbf{M}$ with $|S| = \ell$ and $|T| = d$, there exists a row of $\mathbf{M}$ in which some column in S has a 1 but all columns in T have 0s.*

Proof. For the forward direction, suppose $\mathbf{M}$ is (d, ℓ)-list-disjunct and suppose the conclusion does not how. Let S and T be two disjoint subsets of columns of $\mathbf{M}$ such that $|S| = \ell$, $|T| = d$, and for *every* row i of $\mathbf{M}$ it is *not* true that some column in S has a 1 in row i and all columns in T have 0s in row i. Now, suppose the columns in T correspond to the set of positive items. We claim that the naive decoding algorithm will return every item in $S \cup T$ (plus perhaps some more). But there are $|T| + \ell$ items in $S \cup T$ which violates the definition of (d, ℓ)-list-disjunctiveness. To see the claim, consider any row (i.e. test) i which results in a negative outcome. The naive decoding algorithm eliminates all items which belong to test i. Obviously, no item in T belong to test i otherwise the test is positive. But, no item in S belong to test i either because in row i the columns in T have 0s and no column in S has a 1. Hence, no column in S will be eliminated.

Conversely, suppose the matrix $\mathbf{M}$ satisfies the condition that, for every two disjoint sets S and T of columns of $\mathbf{M}$ with $|S| = \ell$ and $|T| = d$,there exists a row of $\mathbf{M}$ in which some column in S has a 1 but all columns in T have 0s. We will show that $\mathbf{M}$ is (d, ℓ)-list-disjunct which means the naive decoding algorithm always returns a set of items which include all the positives plus $< \ell$ negative items. Let T be the set of positives. Then, $|T| \leq d$. Note that the naive decoding algorithm never eliminate a positive item. Hence, T is included in the output of the naive decoding algorithm. Let $S \cup T$ be the output of the algorithm. We need to show that $|S \cup T| < |T| + \ell$. Suppose for the contrary that $|S \cup T| \geq |T| + \ell$. Then, $|S| \geq \ell$. The reason columns in S were not eliminated is because, for every row i for which the test outcome is negative (i.e. all columns in T have 0s in row i), all columns in S have 0s in row i too! Let S' be an arbitrary subset of ℓ members of $|S|$. Let T' be an arbitrary superset of T such that $|T'| = d$ and $S' \cap T' = \emptyset$. For every row i which columns in T' have 0s in, all columns in T have 0s in, and thus all columns in S have 0s in, which implies all columns in S' have 0s in. Thus, we have exhibited two disjoint subsets S' and T' violating the assumed condition. This is a contradiction. □

Exercise 1. Every (d, ℓ)-list-disjunct matrix is a (d, ℓ)-list-separable matrix. Conversely, every (d, ℓ)-list-separable matrix is a $(d - 1, \ell)$-list-disjunct matrix.

3 Related works

The term *list disjunct* was coined in [11], though it was previously studied under the different names: (d, n, ℓ)-*super-imposed codes* in [5, 7], *list-decoding super-imposed codes* of *strength* d and *list-size* ℓ

in [15][1]. We stick with the name "list disjunct matrices" in these lectures. Shortly prior to [11], Cheraghchi [4] studied the notion of *error-correcting measurement matrices* for d-sparse vectors, which are slightly more general than the notion of list-separable matrices. Since list-separable matrices are equivalent to list-disjunct matrices with a slight loss in parameters, the results in [4], though shown for list-separable matrices, apply to list-disjunct matrices as well. Conversely, our bounds will also apply to error-correcting measurement matrices. We will prove lower bounds which slightly improve lower bounds in [4].

List-disjunct matrices were used in Indyk-Ngo-Rudra [11] to construct efficiently decodable disjunct matrices. They also presented a "stand alone" application of list disjunct matrices in constructing sparsity separators, which were used by Ganguly [8] to design data stream algorithms for the sparsity problem. Rudra and Uurtamo [16] used these objects to design data stream algorithms for tolerant testing Reed-Solomon codes. As observed in De Bonis et. al. [5], these objects can also be used to construct optimal two stage group testing algorithms (where one is allowed to perform tests in two stages, and the second stage of tests can depend on the the first stages results).

List disjunct matrices are also similar to other combinatorial objects such as *selectors* [10] and *multi-user tracing families* [1]. Selectors have found numerous applications such as broadcasting in unknown directed networks and designing tests for coin-weighting problems [10] as well as designing optimal two stage group testing schemes [5]. Multi-user tracing families were used to construct monotone encodings in [2]. Monotone encodings can be used for designing secure vote storage systems [12]. Selectors requirements are slightly stronger than list-disjunct matrices. Both selectors and list-disjunct matrices can be used to recover a super-set of the defective set, while multi-user tracing families are used to recover a sub-set of the defective set. It is not clear if any two of these objects are "equivalent." However, as we shall see, list-disjunct matrices can be used to construct some of these objects.

References

[1] N. ALON AND V. ASODI, *Tracing many users with almost no rate penalty*, IEEE Trans. Inform. Theory, 53 (2007), pp. 437–439.

[2] N. ALON AND R. HOD, *Optimal monotone encodings*, in ICALP '08: Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I, Berlin, Heidelberg, 2008, Springer-Verlag, pp. 258–270.

[3] E. J. CANDÈS, J. K. ROMBERG, AND T. TAO, *Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information*, IEEE Transactions on Information Theory, 52 (2006), pp. 489–509.

[4] M. CHERAGHCHI, *Noise-resilient group testing: Limitations and constructions*, in FCT, 2009, pp. 62–73.

[5] A. DE BONIS, L. GĄSIENIEC, AND U. VACCARO, *Optimal two-stage algorithms for group testing problems*, SIAM J. Comput., 34 (2005), pp. 1253–1270 (electronic).

[6] D. L. DONOHO, *Compressed sensing*, IEEE Transactions on Information Theory, 52 (2006), pp. 1289–1306.

[7] A. G. D′YACHKOV AND V. V. RYKOV, *A survey of superimposed code theory*, Problems Control Inform. Theory/Problemy Upravlen. Teor. Inform., 12 (1983), pp. 229–242.

[8] S. GANGULY, *Data stream algorithms via expander graphs*, in ISAAC, 2008, pp. 52–63.

[9] A. C. GILBERT, Y. LI, E. PORAT, AND M. J. STRAUSS, *Approximate sparse recovery: optimizing time and measurements*, in STOC, 2010, pp. 475–484.

[10] P. INDYK, *Explicit constructions of selectors and related combinatorial structures, with applications*, in SODA, 2002, pp. 697–704.

[1] The authors of [11] were not aware of these earlier work until after the paper was published.

[11] P. INDYK, H. Q. NGO, AND A. RUDRA, *Efficiently decodable non-adaptive group testing*, in Proceedings of the Twenty First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'2010), New York, 2010, ACM, pp. 1126–1142.

[12] T. MORAN, M. NAOR, AND G. SEGEV, *Deterministic history-independent strategies for storing information on write-once memories*, Theory of Computing, 5 (2009), pp. 43–67.

[13] 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.

[14] E. PORAT AND A. ROTHSCHILD, *Explicit non-adaptive combinatorial group testing schemes*, in Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP), 2008, pp. 748–759.

[15] A. M. RASHAD, *Random coding bounds on the rate for list-decoding superimposed codes*, Problems Control Inform. Theory/Problemy Upravlen. Teor. Inform., 19 (1990), pp. 141–149.

[16] A. RUDRA AND S. UURTAMO, *Data stream algorithms for codeword testing*, in ICALP (1), 2010, pp. 629–640.

[17] M. SUDAN, *List decoding: algorithms and applications*, SIGACT News, 31 (2000), pp. 16–27.