Efficiently decodable list-disjunct matrices from list-recoverable codes

The method described here is from Ngo-Porat-Rudra [2], with some basic ideas already appeared in Indyk-Ngo-Rudra [1].

1 List Recoverable Codes

The usual decoding problem is the following: given a received word $y$ which is not necessarily a codeword, recover a near-by codeword $c$. For example, if $y = \text{comtemant}$ we might want to recover $c = \text{complement}$. See Figure 1 for an illustration. In many cases, if we relax the unique decoding requirement, allowing the decoding algorithm to produce a small list of possible codewords, we will be able to design codes with a better rate/distance tradeoff. This is the list decoding problem, illustrated in Figure 1. For example, if $y = \text{complment}$ then we might want to recover the list $\{ \text{complement, complment} \}$.

Generalizing the problem definition further, we consider the notion of list recoverable codes. In the list recovery problem, the input contains for each position $i \in [n]$ has a (small) set $S_i$ of characters. We want to
return a list of codewords agreeing with a large fraction of the sets. For example,

$$\begin{bmatrix}
\{c, f\} \\
\{a, o\} \\
\{t, r\} \\
\{b, h\} \\
\{e, s\} \\
\{a, r\}
\end{bmatrix} \implies \begin{bmatrix}
f \\
a \\
t \\
h \\
e \\
r
\end{bmatrix}, \begin{bmatrix}
m \\
o \\
t \\
h \\
e \\
r
\end{bmatrix}$$

Formally, Let \(\ell, L \geq 1\) be integers and let \(0 \leq \alpha \leq 1\). A \(q\)-ary code \(C\) of block length \(n\) is called an \((\alpha, \ell, L)\)-list recoverable if for every sequence of subsets \(S_1, \ldots, S_n\) such that \(|S_i| \leq \ell\) for every \(i \in [n]\), there exists at most \(L\) codewords \(c = (c_1, \ldots, c_n)\) such that for at least \(\alpha n\) positions \(i, c_i \in S_i\). A \((1, \ell, L)\)-list recoverable code will be henceforth referred to as an \((\ell, L)\)-zero error list recoverable code. We will need the following powerful result due to Parvaresh and Vardy [3]:

**Theorem 1.1** ([3]). For all integers \(s \geq 1\), for all prime powers \(r\) and all powers \(q\) of \(r\), every pair of integers \(1 < k \leq n \leq q\), there is an explicit \(\mathbb{F}_r\)-linear map \(E: \mathbb{F}_q^k \to \mathbb{F}_q^n\) such that:

1. The image of \(E, C \subseteq \mathbb{F}_q^n\), is a code of minimum distance at least \(n - k + 1\).

2. Provided

\[
\alpha > (s + 1)(k/n)^{s/(s+1)}\ell^{1/(s+1)},
\]

\(C\) is an \((\alpha, \ell, O((rs)^s n\ell/k))\)-list recoverable code. Further, a list recovery algorithm exists that runs in \(\text{poly}((rs)^s, q, \ell)\)-time.

We will mostly use the above theorem for the \(r = 2\) case. Let us re-state the special case when \(r = 2\). When \(s = 1\), the code is the Reed-Solomon code.

**Theorem 1.2.** For all positive integers \(s \geq 1\), \(q = 2^m\), \(1 < k \leq n \leq q\), there exists an explicit \(\mathbb{F}_2\)-linear map \(E: \mathbb{F}_2^k \to \mathbb{F}_2^n\) such that:
1. The image \( C \subseteq \mathbb{F}_{2^{2ms}}^n \) of \( E \) is a code of minimum distance at least \( n - k + 1 \).

2. Provided

\[ \alpha > (s + 1)(k/n)^{s/(s+1)}\ell^{1/(s+1)}, \]  \hspace{1cm} (2)

\( C \) is an \( (\alpha, \ell, O(s^s n\ell/k)) \)-list recoverable code. Further, a list recovery algorithm exists that runs in \( \text{poly}(s^s, q, \ell) \)-time.

3. When \( s = 1 \), the code is the RS-code which is \( (\alpha, \ell, O(n\ell/k)) \)-list-recoverable in time \( \text{poly}(q, \ell) \) as long as

\[ \alpha > \sqrt{k\ell/n}. \]  \hspace{1cm} (3)

2 Construct a efficiently decodable list-disjunct matrices from list-recoverable codes

We introduce the idea of using list-recoverable codes to construct efficiently decodable list-disjunct matrices by applying the RS case of the above theorem.

Let \( C_{\text{out}} \) be the \([n, k]_q\)-RS code for \( q \) some power of 2. Let \( C_{\text{in}} \) be any \((d, d)\)-list-disjunct matrix with \( q \) columns and \( t_{\text{in}} \) rows. We have shown using the probabilistic method that there exist \((d, d)\)-list-disjunct matrices with \( q \) columns and \( t_{\text{in}} = O(d \log(q/d)) \) rows. Let \( M = C_{\text{out}} \circ C_{\text{in}} \). We claim that \( M \) is a list-separable matrix which can be efficiently decoded. The decoding algorithm works as follows. (See Figure 3 for an illustration.)

- From the \( t_{\text{in}} \) test results for each position \( i \in [n] \), we run the naive decoding algorithm for \( C_{\text{in}} \) to recover a set \( S_i \) of less than \( \ell = 2d \) columns of \( C_{\text{in}} \).
- These columns naturally correspond to a set \( S_i \) (overloading notation) of symbols of the outer code.
As long as $1 > k\ell/n$, Theorem 1.2 ensures that there is a poly$(q, \ell)$-time algorithm which recovers a list of $L = O(n\ell/k)$ codewords of $C_{\text{out}}$ each of which agrees with all the $S_i$. These codewords certainly contain all of the positives. (Why?)

To minimize the number of tests, which is $O(n \cdot t_{\text{in}}) = O(nd\log(q/d))$, we can choose the parameters as follows.

$$n = q$$
$$q = \frac{4d\log N}{\log(4d\log N)}$$
$$k = \frac{\log N}{\log q}.$$

We need to verify that $k\ell < n$ which is the same as $2d\log N < q\log q$. Note that

$$q\log q = \frac{4d\log N}{\log(4d\log N)} \log \left( \frac{4d\log N}{\log(4d\log N)} \right) = (2d\log N) \cdot 2 \left( 1 - \frac{\log\log(4d\log N)}{\log(4d\log N)} \right) > 2d\log N$$

whenever

$$\frac{\log\log(4d\log N)}{\log(4d\log N)} < 1/2.$$

But the above holds true for any $d \geq 1, N \geq 3$. The total number of tests is

$$t = O \left( \frac{4d^2\log N}{\log(4d\log N)} \log \left( \frac{4\log N}{\log(4d\log N)} \right) \right) = O(d^2 \log N).$$

The total decoding time is $O(nqt_{\text{in}} + \text{poly}(q, \ell)) = \text{poly}(t)$. Stacking this efficiently decodable $(d, L)$-list-separable matrix on top of any $d$-disjunct matrix, and we obtain an efficiently decodable $d$-disjunct matrix with the best known number of tests. We just proved the following theorem.

**Theorem 2.1.** By concatenating the RS-code with a good list-disjunct inner code (i.e. matrix), we obtain a $(d, L)$-list-disjunct matrix with $L = O(d^2)$ which is decodable in time $\text{poly}(d, \log N)$. The total number of tests is $O(d^2 \log N)$. Thus, by stacking the result on top of a $d$-disjunct matrix with $O(d^2 \log N)$, we obtain a $d$-disjunct matrix with $t = O(d^2 \log N)$ rows which is decodable in $\text{poly}(t)$-time.

Since we do not know of any way to construct explicit (or strongly explicit) $(d, d)$-list-disjunct matrices, the above construction is not explicit. Of the three objectives: (1) minimum number of tests, (2) explicitly constructible, (3) fast decoding, the above construction gives us (1) and (3) but not (2).

**Open Problem 2.2.** Find a (strongly or not) explicit construction of $(d, d)$-list-disjunct matrices attaining the probabilistic bound $O(d\log(N/d))$.

Some application does not require disjunct matrix, but only a $(d, \text{poly}(d))$-list-disjunct matrix which is efficiently decodable. From the above, we are able to construct from the RS-code an efficiently decodable $(d, \Theta(d^2))$-list-disjunct matrix with $t = O(d^2 \log N)$ number of rows. However, the probabilistic bound for $(d, \Omega(d))$-list-disjunct matrices says that we can achieve $t = O(d\log(N/d))$ rows. Thus, there is still work to be done here too.

**Open Problem 2.3.** Find a (strongly or not) explicit construction of $(d, \text{poly}(d))$-list-disjunct matrices attaining the probabilistic bound $O(d\log(N/d))$ and are efficiently decodable.

In the next section, we will use the PV* code instead of the RS = PV code to show that we can partly address this problem.
3 Construct a efficiently decodable list-disjunct matrices from $PV^s$ codes

In this section, we prove a generic lemma where the outer code is the $PV^s$-code and the inner code is an arbitrary $(d, \ell)$-list-disjunct matrix. Later we shall apply the lemma by “plugging-in” different values of $s$ and different constructions of $(d, \ell)$-list-disjunct matrices. What is interesting about this lemma is that it shows a black-box conversion procedure which converts a (family of) list-disjunct matrix into another one which is efficiently decodable.

**Lemma 3.1** (Black-box conversion using list-recoverable codes). Let $\ell, d \geq 1$ be integers. Assume that for every $Q \geq d$ there exists a $(d, \ell)$-list-disjunct matrix with $\bar{\ell}(d, \ell, Q)$ rows and $Q$ columns. For all integers $s \geq 1$ and $N \geq d$, define

$$A(d, \ell, s) = (d + 1)^{1/s}(s + 1)^{1+1/s}.$$  

Let $k$ be the minimum integer such that $k \log(kA(d, \ell, s)) \geq \log N$, and $q = 2^m$ be the minimum power of 2 such that $q > kA(d, \ell, s)$. Then, there exists a $(d, L)$-list separable $t \times N$ matrix $M$ with the following properties:

(i) $t = O\left(s^{1+1/s} \cdot (d + \ell)^{1/s} \cdot \left(\frac{\log N}{\log q}\right) \cdot \bar{\ell}(d, \ell, q^s)\right)$.

(ii) $L = s^{O(s)} \cdot (d + \ell)^{1+1/s}$.

(iii) It is decodable in time $t^{O(s)}$.

Furthermore, if the $\bar{\ell}(d, \ell, Q) \times Q$ matrix is (strongly) explicit then $M$ is (strongly) explicit.

**Proof.** Let $M$ be the concatenation of $C_{out} = PV^s$ with $C_{in}$ which is a $(d, \ell)$-list-disjunct matrix with $\bar{\ell}(d, \ell, Q)$ rows and $Q = q^s$ columns. (Recall that the $PV^s$-code has length $n$, alphabet size $q^s = 2^{ms}$, and $q^k$ codewords.) We will have to choose parameters $1 < k \leq n \leq q$ so that the $PV^s$-code is $(\alpha = 1, d + \ell, O(s^s n (d + \ell)/k))$-list-recoverable. In particular, the followings must hold:

$$N \leq q^k$$

(because there are $q^k$ codewords)

$$1 > (s + 1)^{s+1} (k/n)^{s(d + \ell)}$$

(to satisfy (1) with $\alpha = 1$).

We will pick $q = n$ and $k$ such that $\log N \leq k \log q = k \log n$. The second condition is satisfied iff $q = n > k(s + 1)^{1+1/s}(d + \ell)^{1/s} = kA(d, \ell, s)$. Hence, if $q$ and $k$ satisfy the conditions stated in the lemma then the above two inequalities are satisfied.

The number of rows of $M$ is

$$t = n \cdot \bar{\ell}(d, \ell, Q) \leq 2kA(d, \ell, s)\bar{\ell}(d, \ell, Q) = O\left(\frac{\log N}{\log(kA(d, \ell, s))}\right) A(d, \ell, s)\bar{\ell}(d, \ell, Q) = O\left(\frac{\log N}{\log(q/2)}\right) A(d, \ell, s)\bar{\ell}(d, \ell, Q) = O\left(\frac{\log N}{\log q}\right) A(d, \ell, s)\bar{\ell}(d, \ell, Q).$$

To show the matrix is list-separable we use the natural decoding algorithm which is identical to the one we did for the RS-code in the previous section. First, we run the naive decoding algorithm for each position.
$i \in [n]$ to obtain a list of less than $d + \ell$ columns of the inner code. Naturally, the column list for each position $i$ corresponds to a set $S_i$ of size less than $d + \ell$. Finally, we run the list-recovery algorithm for the $PV^s$ outer code to obtain a list of at most $L = O(s^n(d + \ell)/k) = O(s^s(s + 1)^{1+1/s}(d + \ell)^{1+1/s})$ codewords.

Now, fix any constant $0 < \epsilon < 1$ and $s = 1/\epsilon$. We apply the above lemma with a random inner code which is $(d, d)$-list-disjunct with $t = O(d \log(q^s/d)) = O(ds \log(q))$ rows and $q^s$ columns. Then, we obtain an efficiently decodable $(d, (1/\epsilon)^{O(1/\epsilon)}d^{1+\epsilon})$-list-separable matrix $M$ with: $t = O((1/\epsilon)^{2+\epsilon}d^{1+\epsilon}\log N)$ rows, $N$ columns. That is a proof of the following simple corollary.

**Corollary 3.2** (Concatenating $PV^s$ with a random inner code). For every $\epsilon > 0$, there exists an efficiently decodable $(d, (1/\epsilon)^{O(1/\epsilon)}d^{1+\epsilon})$-list-disjunct matrix $M$ with $N$ columns and $t = O((1/\epsilon)^{2+\epsilon}d^{1+\epsilon}\log N)$ rows.

**References**

