Error Correcting Codes: Combinatorics, Algorithms and Applications (Spring 2010)

Lecture 41: Group Testing and List Recovery

April 23, 2010

Lecturer: Atri Rudra *Scribe: Jiun-Jie Wang*

In the last lecture, we proved that the cormode-muthukrishnan algorithm for the hot item problems can match the following data stream requirement: one-pass, poly-log space and update time. In this lecture, we will show that reporting time of hot item problem can also be done in poly-log time. Recall that we use the concept of concatenation code to build a d-disjunct matrix. Last lecture, we saw that the reporting problem is the same as decoding for the group testing problem: given $\mathbf{r}, M$, when we know $\mathbf{r} = M\mathbf{x}$, output $\mathbf{x}$. The decoding time for a general d-disjunct matrix as we have seen is $O(nt)$. It doesn't match our requirement. So, we need a more smart decoding algorithm for hot items problem.

1 Naive Idea

In the last lecture, we used M_{C^*} as the d-disjunct matrix, when $C^* = C_{out} \circ C_{in}$, C_{out}: $[q,k]_q$ RS Code. $C_{in} : [q] \to \{0,1\}^q$ is the identity code.

Example 1: Let $k = 1$, $q = 3$, $C_{out}(\mathbf{m}_0) = (0,0,0), C_{out}(mathbfm_1) = (1,1,1), C_{out}(\mathbf{m}_2) = (2,2,2)$, $n = q^k = 3$, $t = q \times q = 9$,

$$C_{out} = \begin{bmatrix} C_{out}(\mathbf{m}_0)_0 & C_{out}(\mathbf{m}_1)_0 & C_{out}(\mathbf{m}_2)_0 \\ C_{out}(\mathbf{m}_0)_1 & C_{out}(\mathbf{m}_1)_1 & C_{out}(\mathbf{m}_2)_1 \\ C_{out}(\mathbf{m}_0)_2 & C_{out}(\mathbf{m}_1)_2 & C_{out}(\mathbf{m}_2)_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 1 & 2 \end{bmatrix}$$

Now, note that C_{in} behaves as follows: $C_{in}(0) = 001$, $C_{in}(1) = 010$, $C_{in}(2) = 100$

suppose $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$ Then,

$$M_{C^*} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$M_{C^*}\mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{r}$$

In the example above, given $\mathbf{r}$, it is easy to figure out $\mathbf{x}$. However, in general, decoding of $\mathbf{r}$ will follow two step process of decoding concatenated code.

Firstly, C_{out} is a $[q, k]_q$ code. So, the message set of C_{out} has size q^k. From the code construction of C_{out}, we know each message $\mathbf{m}$ of $\mathbb{F}_q^k$ corresponding to a column index of M_{C^*}. Thus, $\mathbb{F}_q^k \approx [n]$. It implies that for each $\mathbf{x}_j$, we can find a message $\mathbf{m}$ s.t. $\mathbf{x}_j = \mathbf{x}_\mathbf{m}$. Secondly, we can also show $\mathbb{F} \times \mathbb{F} \approx [t]$. So, we can have $\mathbf{r}_{j'} = \mathbf{r}_i(j)$ where $j' \in [t]$ and $i, j \in \mathbb{F}_q$.

How to perform decoding of C_{in} from $\mathbf{r}$? If $\mathbf{r}_i(j) = 1$, then c_i is possible to be j. $\mathbf{r}_i(j) = 0$, then c_i can not be j where c_i, i and $j \in \mathbb{F}_q$. Finally, Claim 1 leads us to find answer.

Claim 1. *If* $\mathbf{x}_\mathbf{m} = 1$ *and* $(c_1, \cdots, c_q) = C_{out}(\mathbf{m})$*, then* $c_i \in S_i$*,* $1 \leq i \leq q$ *where*

$$S_i = \{j \in \mathbb{F}_q \mid r_i(j) = 1\}.$$

Proof. Left exercise. □

2 The Decoding Algorithm

ALGO 1:

- Input: $\mathbf{r} = (r_1, r_2, \cdots, r_q) \in (\{0,1\}^q)^q$.

  1. (decoding C_{in}): For every $0 \leq i \leq q-1$, let

  $$S_i = \{j \in \mathbb{F}_q \mid r_i(j) = 1\}.$$

  2. (decoding C_{out}): Run an error-less list recovery algorithm on $S_1, \cdots, S_q$ to get

  $$O = \{\mathbf{m}_1, \cdots, \mathbf{m}_L\}$$

  , then $\widehat{x}_\mathbf{m} = 1$ if $\mathbf{m} \in O$. $\widehat{x}_\mathbf{m} = 0$, otherwise.

Recall that Error-Less List Recovery for $C \subseteq \sum^n$.

- Input: $S_1, \cdots, S_n$ s.t. $S_i \subseteq \sum$.

- Output: All $(c_1, \cdots, c_n) \in C$ s.t. $c_i \in S_i$, $1 \leq i \leq n$.

For a $[n, k]_q$ RS code, we can do error-less list-recovery in polylog time if $l <$????.
After running error-less list-recovery algorithm, we can get the following fact, for a $\mathbf{x}$,

$$\widehat{\mathbf{x}}_{\mathbf{m}} = 1 \text{ implies } \mathbf{x}_{\mathbf{m}} = 1$$

, however, at the same, this algorithm also produces $\mathbf{m} \in O$ to s.t. $\mathbf{x}_{\mathbf{m}} = 0$, ie.

$$\widehat{\mathbf{x}}_{\mathbf{m}} = 0 \nRightarrow \mathbf{x}_{\mathbf{m}} = 0.$$

Thus, for $\mathbf{m}$ s.t. $\widehat{\mathbf{x}}_{\mathbf{m}} = 1$ (but $\mathbf{x}_{\mathbf{m}} = 0$), we need to make $\mathbf{x}_{\mathbf{m}} = 0$.
For this problem, we can run the naive decoding algorithm for a disjunct matrix, but only for columns in O. ALGO 2:

- Input: $O = \{\mathbf{m}_1, \cdots, \mathbf{m}_L\}$.

  1. For all $\mathbf{m} \in O$, set $\widehat{\mathbf{x}}_{\mathbf{m}} = 1$.

  2. For $j = 1 \cdots t$ if $r_j = 0$, then for all $\mathbf{m} \in O$ if $M_{j,\mathbf{m}} = 1$, set $\widehat{\mathbf{x}}_{\mathbf{m}} = 0$.

  3. Output $\mathbf{m} \in O$ if and only if $\widehat{\mathbf{x}}_{\mathbf{m}} = 1$.

Lemma 1. *If ALGO2 is run on the output of ALGO1 , then* $M_{C^*}\hat{\mathbf{x}} = \mathbf{r}$.

Proof. In this proof, we want to remove some $\mathbf{m} \in O$ s.t.

$$M_{C^*}\hat{\mathbf{x}} = \mathbf{r} = M_{C^*}\hat{\mathbf{x}}$$

Let $\hat{\mathbf{r}} = M_{C^*}$ and $j' = pq + j$. We go through this proof case by case

- Case 1: $\mathbf{x}_{\mathbf{m}} = 0$, $C_{in}(C_{out}(\mathbf{m})_p)_j = 0$ and $\mathbf{r}_{j'} = 0$.
  If we set $\hat{\mathbf{x}}_{\mathbf{m}} = 1$, we can not change $\hat{\mathbf{r}}_j$ from 0 to 1.

- Case 2: $\mathbf{x}_{\mathbf{m}} = 0$, $C_{in}(C_{out}(\mathbf{m})_p)_j = 1$ and $\mathbf{r}_{j'} = 0$.
  If we set $\hat{\mathbf{x}}_{\mathbf{m}} = 1$, we will change $\hat{\mathbf{r}}_j$ from 0 to 1.

So, by case 1 and case 2, we can conclude that ALGO2 output $\mathbf{m} \in O$ s.t. $\mathbf{x}_{\mathbf{m}} = 1$. □

Theorem 1. *The updating step of hot item problem can be done in* $O(t \times$ *poly-log*$(tn))$ *time.*

Proof. ALGO 1 runs O(poly-log(tn)) time and ALGO 2 runs $O(t \times L)$ time.
We know L is O(poly-log(tn)). It implies that total time is $O(t \times$ poly-log$(tn))$ time. □