Error Correcting Codes: Combinatorics, Algorithms and Applications **(Fall 2007)**

Lecture 37: List Decoding of RS

November 27, 2007

Lecturer: Atri Rudra *Scribe: Thanh-Nhan Nguyen & Atri Rudra*

1 List Decoding Recap

Recall that list decoding capacity is $1-H_q(p)$. In particular, we know that there exists an $(p, O(\frac{1}{\varepsilon}))$-list decodable code of rate $1 - H_q(p) - \varepsilon$. Here are some natural follow-up questions.

Question 1.1. *Are there explicit codes that achieve list decoding capacity of* $1-H_q(p)$ *with efficient list decoding algorithms ?*

Recall that $1 - H_q(p) = 1 - p - \varepsilon$ for $q = 2^{\Omega(\frac{1}{\varepsilon})}$, which leads to the following question:

Question 1.2. *Are there explicit codes with rate* $R > 0$ *that can list-decodable up to* $1 - R - \varepsilon$ *faction of errors with efficient list decoding algorithms?*

Recall that the alphabet-free version of the Johnson bound states that any code with relative distance δ is $(J_q(\delta), O(n^2))$-list decodable, where

$$J_q(\delta) = 1 - \sqrt{1 - \delta}.$$

Further, by the Singleton bound,

$$1 - \delta \geq R - o(1),$$

which in turn implies that

$$J_q(\delta) \leq 1 - \sqrt{R}.$$

This leads to the following question:

Question 1.3. *Is there an efficient list decoding algorithm for code of rate* $R > 0$ *that can correct* $1 - \sqrt{R}$ *fraction of errors?*

Note that in the question above, explicitness is not an issue as e.g., a Reed-Solomon code of rate R by the Johnson bound is $(1 - \sqrt{R}, O(n^2))$-list decodable.

We will begin with trying to answer the question above. We will study an efficient list decoding algorithm for Reed-Solomon codes that can correct up to $1 - \sqrt{R}$ fraction of errors. To this end, we will present a sequence of algorithms for (list) decoding Reed-Solomon codes that ultimately will answer Question 1.3.

2 Decoding Reed-Solomon Codes

Consider any $[n, k+1]_q$ RS code that has the evaluation set $\{\alpha_1, \ldots, \alpha_m\}$. Below is a formal definition of the decoding problem for RS codes:

- Input: Received word $(y_i)_{i=1}^n$, $y_i \in \mathbb{F}_q$ and error parameter $e = n - t$.
- Output: All polynomials $P(X) \in \mathbb{F}_q[X]$ of degree at most k such that $P(\alpha_i) = y_i$ for at least t values of i.
- Goal: Make t as small as possible.

2.1 Berlekamp Welch Algorithm

We begin with the unique decoding regime, where $t > \frac{n+k}{2}$. We looked at the Berlekamp-Welch algorithm:

- **Step 1:** Find polynomials $B(X)$ of degree $k+e$, and $N(X)$ of degree e such that

$$\forall 1 \leq i \leq n, B(\alpha_i) = y_i N(\alpha_i).$$

- **Step 2:** If $Y - P(X)$ divides $Q(X, Y) = YE(X) - B(X)$, then output $P(X)$ (assuming $\Delta(\mathbf{y}, (P(\alpha_i))_{i=1}^n) \leq e$).

2.2 Structure of list decoding algorithms for RS

The Berlekamp-Welch Algorithm has the following structure:

- **Step 1:** (Interpolation Step) Find non-zero $Q(X, Y)$ such that $Q(\alpha_i, y_i) = 0, 1 \leq i \leq n$.
- **Step 2:** (Root Finding Step) If $Y - P(X)$ is a factor of $Q(X, Y)$, then output $P(X)$ (assuming it is close enough to the received word).

For the Berlekamp-Welch algorithm, **Step 2** is easy to implement. For the general case, it is known that factoring bivariate polynomials can be done in polynomial time (see e.g. [1]). All the list decoding algorithms that we will see in this course, will follow the above two step algorithm skeleton. In particular, for RS codes, we will need to ensure the following:

- **Step 1** requires solving for the co-efficients of $Q(X, Y)$. This can be done as long as the number of coefficients is greater than the the number of constraints.
- For **Step 2**, to ensure that for every polynomial $P(X)$ that needs to be output $Y - P(X)$ divides $Q(X, Y)$, we will add restrictions on $Q(X, Y)$.

First, we recall the definition of maximum degree of a variable.

Definition 2.1. *$deg_X(Q)$ is the maximum degree of X in $Q(X,Y)$. Similarly, $deg_Y(Q)$ is the maximum degree of Y in $Q(X,Y)$*

Given $deg_X(Q) = a$ and $deg_Y(Q) = b$, we can write

$$Q(X,Y) = \sum_{\substack{0 \le i \le a, \\ 0 \le j \le b}} q_{ij} X^i Y^j,$$

where the coefficients $q_{ij} \in \mathbb{F}_q$. Note that the number of coefficients is equal to $(a+1)(b+1)$. (We want this product greater than n.)

2.3 Algorithm 1

Below is a list decoding algorithm that generalizes the Berlekamp-Welch algorithm (and follows the two step skeleton outlined above).

- **Step 1:** Find a non-zero $Q(X,Y)$ with $deg_X(Q) \le \ell, deg_X(Q) \le \frac{n}{\ell}$ for some ℓ to be fixed later such that
$$Q(\alpha_i, y_i) = 0, 1 \le i \le n.$$
- **Step 2:** Output $P(X)$ if $Y - P(X)$ divides $Q(X,Y)$ (and $\Delta(\mathbf{y}, (P(\alpha_i))_{i=1}^n) \le e$).

To ensure the correctness of **Step 1**, we will need to ensure that the number of coefficients (which is $(\ell+1)(n/\ell+1)$) is larger than the number of constraints (which is n). Indeed,

$$(\ell+1)(\frac{n}{\ell}+1) > \ell \cdot \frac{n}{\ell} = n.$$

For **Step 2**, we need to show that if $P(X)$ of degree $\le k$ agrees with Y in at least t positions, then $Y - P(X)$ divides $Q(X,Y)$. Towards this end, we define

$$R(X) \triangleq Q(X, P(X)).$$

Note that we need to show $R(X) \equiv 0$. For the sake of contradiction, assume that $R(X) \not\equiv 0$. Note that

$$\begin{aligned} deg(R) &\le deg_X(Q) + deg(P) \cdot deg_Y(Q) && (1) \\ &\le \ell + \frac{nk}{\ell}. && (2) \end{aligned}$$

On the other hand, if $P(\alpha_i) = y_i$ then

$$Q(\alpha_i, y_i) = Q(\alpha_i, P(\alpha_i)) = 0.$$

Thus, α_i is a root of $R(X)$. In other words R has at least t roots. Note that this will lead to a contradiction if $t > deg(R)$, which will be true if

$$t > \ell + \frac{nk}{\ell}.$$

If we pick $\ell = \sqrt{nk}$, we will have $t > 2\sqrt{nk}$. Thus,

Theorem 2.2. *Algorithm* 1 *for RS codes of rate* R*, can list decode from* $1-2\sqrt{R}$ *fraction of errors. Further, the algorithm can be implemented in polynomial time.*

The claim on the efficient run time follows as **Step 1** can be implemented by Gaussian elimination and **Step 2** can be computed using e.g. the algorithm from [1].

References

[1] E. Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM J. Comput.*, 14(2):469–489, 1985.