ENEE626: Error-Correcting Codes

Instructor: Alexander Barg — Notes by: Jing Yang

Lecture 19 (11/10/05). Sudan's algorithm for RS decoding

List decoding of error-correcting codes — http://www.ece.umd.edu/~abarg/626

Recall from last lecture:

RS codes as evaluation codes: Let $\mathbb{F}_q(x)$ be a finite field and $(x_1, x_2, ..., x_n) \subset \mathbb{F}_q$ be n field elements (points).

By definition, $\mathbf{c} = \text{eval}(f) \Leftrightarrow \mathbf{c} = (f(x_1), f(x_2), ..., f(x_n))$

A q-ary $RS[n, k]$ code C is

$$\{\mathbf{c} = \text{eval}(f); f \in \mathbb{F}_q[x], \deg f \leq k-1\}$$

Berlekamp-Welch algorithm for RS decoding Let $t = \lfloor \frac{n-k}{2} \rfloor$; let $\mathbf{r} = (r_1, r_2, ...r_n)$ be the received vector. Note that if there is no error in coordinate i then $r_i = c_i = f(x_i)$.

1. Solve the system $Q(x_i, r_i) = 0$, $i = 1, 2, ..., n)$ with respect to the coefficients $Q_{i,j}$. More concretely, the system has the form

$$(\mathbf{Q}, \mathbf{X}_i) = 0 \quad (1 \leq i \leq n)$$

where $\mathbf{Q} = (Q_{0,0}, Q_{0,1}, ...Q_{0,n-1-t}; Q_{1,0}, Q_{1,1}, ...Q_{n-1-t-(k-1)})$
and $\mathbf{X}_i = (x_i^0, x_i^1, ..., x_i^{n-1-t}; r_i x_i^0, r_i x_i^1, ..., r_i x_i^{n+t-(k-1)})$

2. Find $f(x) = -\frac{Q_0(x)}{Q_1(x)}$.

3. Decode to $\mathbf{c} = \text{eval}(f)$.

Theorem: Let $\mathbf{c} = \text{eval}(f)$ and let $\mathbf{r} = \mathbf{c} + \mathbf{e}$ be the received vector, where $\text{wt}(\mathbf{e}) \leq t$ is the error vector. Let $Q(x, y) = Q_0(x) + yQ_1(x) \neq 0$. $\deg Q_0 \leq n - 1 - t$, $\deg Q_1 = n - 1 - t - (k-1)$ be a polynomial such that $Q(x_i, r_i) = 0, i = 1, 2, ..., n$. Then $f = -\frac{Q_0}{Q_1}$.

If $Q(x, y)$ satisfies $Q(x, f(x)) = 0$, then $f(x)$ is called a y-root of Q and $(y - f(x))|Q(x, y)$.

Remark: By the construction we have $Q(x, y) = Q_0(x) + yQ_1(x) = Q_1(x)(y + Q_0(x)/Q_1(x)) = Q_1(x)(y - f(x))$. The input of the algorithm is (x_i, r_i). If there is no error then $r_i - f(x_i) = 0$, If there is an error, then $r_i - f(x_i) \neq 0$, but $Q(x_i, r_i) = 0$, so $Q_1(x_i) = 0$. Therefore $Q_1(x)$ is a multiple of the error locator polynomial. It is possible to write the GPZ algorithm in the form similar to BW, see the book of Justesen and Høholdt pp.57-60.

Sudan's algorithm

Preliminaries: list decoding. If $\tau \leq t = \lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{n-k}{2} \rfloor$, then for all $\mathbf{r} \in \mathbb{F}_q^n$, the ball of radius τ centered at $\mathbf{r}$ contains at most one code point: $|B_\tau(\mathbf{r} \bigcap C)| \leq 1$
If $\tau > t$, then this is no longer true. There may exist several different codewords within distance τ of a received word.

Definition: List decoding of an error-correcting code C is a procedure to recover all codewords of C at $distance \leq \tau$ from the received vector $\mathbf{r}$. List decoding is considered correct if the transmitted vector $\subseteq$ is in the list. If this holds true for some decoding algorithm, we will say that the algorithm corrects up to τ errors in the list-decoding sense.

$C[n,k,n-k+1]$, $\mathbf{c}=\mathrm{eval}(f)$, $\deg f \le k-1$; $\mathbf{r}=\mathbf{c}+\mathbf{e}$, $\mathrm{wt}(\mathbf{e})\le\tau$.
Let $Q(x,y)=Q_0(x)+Q(x)y+Q_2(x)y^2+...+Q_l(x)y^l$ be such that
1. $Q(x_i,r_i)=0$, $1\le i\le n$.
2. $\deg(Q_j(x))\le n-\tau-1-j(k-1)$, $j=0,1,...l$.

Lemma: If $\mathbf{c}=\mathrm{eval}(f)$ and at $(\mathbf{e})\le\tau$ then $(y-f(x))|Q(x,y)$.
Proof: Consider the polynomial $Q(x,f(x))$. It is a polynomial of one variable, we have $\deg Q(x,f(x))\le n-\tau-1$, so it has at most $n-\tau-1$ zeros. However, $\#\{i: r_i\ne f(x_i)\}\le\tau$ i.e.$r_i=f(x_i)$ for at least $n-\tau$ values of i. Since $Q(x_i,r_i)=0, 1\le i\le n$ we observe that $Q(x_i,f(x_i))=0$ for $n-\tau$ or more values of i. Therefore So, $Q(x,f(x))=0$ meaning that f is a y-root of Q. ∎
Since $\deg Q\le l$, the polynomial $Q(x,y)$ has at most l y-roots: $f_1,f_2,...f_l$. This means that there will be no more than l codewords in the list.

Algorithm: input τ,l,C,$\mathbb{F}_q$,$\mathbf{r}$
1. Solve the linear system $Q(x_i,r_i)=0$. with respect to the coefficients of Q.
2. Find y-roots of Q: f_1,f_2,...,f_l.
Check that the codewords $\mathbf{c}_j=\mathrm{eval}(f_j)$ satisfy $d(\mathbf{c}_j,\mathbf{r})\le\tau$; discard those that do not and output the remaining list.

Next question: under which condition on the parameters τ and l does the system $Q(x_i,r_i)=0$ have a nonzero solution?
Regard all the coefficients of $Q(x,y)$ as unknowns, then from the definition of $Q(x,y)$, we have n linear equations for these unknowns. On the other hand, the number of coefficients of the polynomial Q is

$$(n-\tau)(l+1)-\sum_{j=1}^{l}j(k+1)=(n-\tau)(l+1)-(k-1)\frac{l(l+1)}{2}$$

Thus if $(l+1)(n-\tau)-(k-1)\frac{l(l+1)}{2}>n$, there are more unknowns than constraining equations. Since we are dealing with a homogeneous system of linear equations, it is always possible to find a solution $Q\not\equiv 0$. We obtain the condition

$$-\tau(l+1)>n-n(l+1)+(k-1)\frac{l(l+1)}{2}$$

or

$$\tau<\frac{ln}{l+1}-(k-1)\frac{l}{2} \tag{1}$$

At the same time, we must also require that the degree of the polynomials Q_j be nonnegative. The most stringent condition is obtained for $j=l$ and has the form

$$(n-\tau)-l(k-1)>0,$$

or

$$\tau<n-l(k-1).$$

Before analyzing these two conditions in general let us examine the cases of a small list size, $l=1,2$.
$l=1\Rightarrow\tau<\frac{n-k+1}{2}$, so we have recovered the Berlekamp-Welch decoding
$l=2\Rightarrow\tau<2/3n-(k-1)$ and $\tau<n-2(k-1)$.
Suppose that $n-2(k-1)<2/3n-(k-1)$ or $n/3-k+1<0$ or $k/n>1/3+1/n$. We see that if $\frac{k}{n}<\frac{1}{3}+\frac{1}{n}$, then $\tau<\frac{2}{3}n-(k-1)$ or $\frac{\tau}{n}<-\frac{k}{n}+\frac{2}{3}+\frac{1}{n}$.
If $\frac{k}{n}<\frac{1}{3}+\frac{1}{n}$, then the condition $\tau<n-2(k-1)$ is more restrictive than the other one, meaning that the algorithm does not even reach the $d/2$ decoding radius.

Generally, to ensure that the algorithm decodes beyond half the minimum distance we have to compare the decoding radius from (1) to $d/2$. We obtain the inequality

$$\frac{l}{l+1}n-(k-1)\frac{l}{2}>\frac{n-k+1}{2}$$

which implies the following condition for the code rate:

$$\frac{k}{n}<\frac{1}{l+1}+\frac{1}{n}.$$

From (1) we get

$$\frac{\tau}{n}<-\frac{l}{2}\frac{k}{n}+\frac{l}{l+1}+\frac{l}{2n}$$

Thus we obtain
Claim: If $\frac{k}{n}<\frac{1}{l+1}+\frac{1}{n}$, then the algorithm corrects up to

$$\frac{\tau}{n}<-\frac{l}{2}\frac{k}{n}+\frac{l}{l+1}+\frac{l}{2n}$$

relative proportion of errors.

Finally, let us estimate the *asymptotic error-correction radius* τ of the Sudan algorithm as the list size $l\to\infty$. There are two conditions on the parameters of the algorithm.

Condition 1: $\deg Q_j>0$, for all $j=0,1,...l$ or $\deg Q_j=n-\tau-1-j(k-1)$ As remarked above, the most stringent of these is the condition for $j=l$:

$$\tau<n-l(k-1). \tag{2}$$

Let us take τ the maximum possible, i.e.,

$$n-\tau=l(k-1)-1. \tag{3}$$

Condition 2: If

$$\tau+(k-1)\frac{l}{2}<\frac{ln}{l+1} \tag{4}$$

then the system $Q(x_i,r_i)=0$ has a nonzero solution for the coefficients $Q_{j,k}$.

Substituting (3) into (4), we obtain

$$n-l(k-1)+(k-1)\frac{l}{2}-1=n-1-\frac{l}{2}(k-1)<\frac{ln}{l+1}.$$

This is equivalent to the inequalities

$$(n-1)(l+1)-\frac{l(l+1)(k-1)}{2}<ln$$

$$nl+n-(l+1)-\frac{l(l+1)(k-1)}{2}<ln$$

$$2n<l^2(k-1)+l(k-1)+2(l+1). \tag{5}$$

For the inequality (5) to be true it is sufficient that the following inequality hold true

$$2n<l^2(k-1).$$

This is equivalent to

$$l(k-1) > \sqrt{2n(k-1)}.$$

Substituting this into (2) we obtain

$$\tau \leq n - \sqrt{2n(k-1)} = n(1 - \sqrt{2R - 2/n})$$

If this inequality is fulfilled, then conditions 1-2 are satisfied.

For large n, the condition on τ/n becomes

$$\frac{\tau}{n} \lesssim 1 - \sqrt{2R}$$

where R is the code rate.

The error correction radii of the Sudan algorithm are plotted vs. the code rate $R = k/n$ in Figure 1.

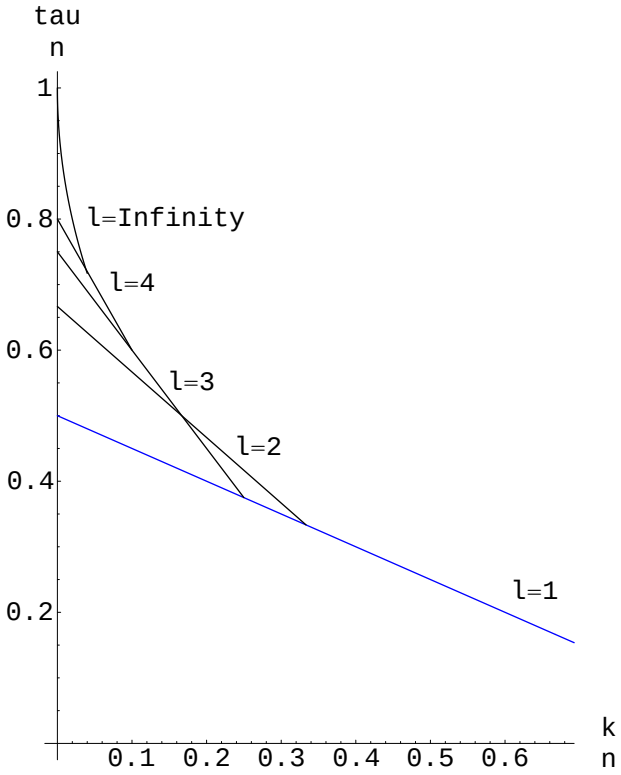

Figure 1: The relative error-correction radius τ/n of the Sudan algorithm vs the code rate k/n for the list size $l = 1, 2, 3, 4, \infty$.