U.C. Berkeley — CS294: Codes and Complexity
Professor Luca Trevisan

Handout N6
Scribe: Hoeteck Wee

9/15/2003

# Notes for Lecture 6

## 6.1 Outline

Today's lecture will cover:

- List-Decoding for Reed-Solomon Codes
- List-Decoding for Concatenated Codes (Reed-Solomon with Hadamard)

## 6.2 List-Decoding of Reed-Solomon Codes

Recall that the $[n, k, n-k+1]_q$ Reed-Solomon Code regards a message as a polynomial $p$ of degree $k-1$, and the encoding is given by $(p(x_1), \ldots, p(x_n))$, where $x_1, \ldots, x_n$ are distinct elements of a field $\mathcal{F}_q$ of size $q$. Upon passing through a noisy channel, we receive $y_1, y_2, \ldots, y_n$. Suppose we are told that we have agreement $t$, that is $\#i : p(x_i) = y_i \geq t$. Then,

1. If $t \geq \frac{n+k}{2}$, we have an efficient algorithm for unique decoding to recover $p$ from $(y_1, \ldots, y_n)$. In particular, we can uniquely decode from $n(1/2 + k/2n)$ agreement, or a fractional agreement that is arbitrarily close to $1/2$ if we fix $k$ and increase $n$. Note that the condition $t \geq \frac{n+k}{2}$ is necessary for unique decoding.
2. If $t > 2\sqrt{nk}$, we shall see that we can still efficiently list-decode from $(y_1, \ldots, y_n)$ to recover a small list of polynomials that includes $p$. This means that we can list-decode from $n(2\sqrt{k/n})$ agreement, or a fractional agreement that is arbitrarily close to 0 if we fix $k$ and increase $n$.

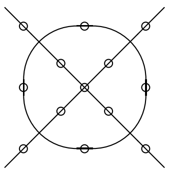
The list-decoding problem for Reed-Solomon codes can be stated as follows: given $n$ distinct points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ in $\mathcal{F}_q^2$ and parameters $k, t$, find a list of all polynomials $p$ such that:

1. $p$ has degree $\leq k-1$; and
2. $\# : p(x_i) = y_i \geq t$

With no further constraints on $n, k$ and $t$, it is not clear that the list of such polynomials is small (that is, $\text{poly}(n, k, t)$). In particular, if $t = k$, there are at least $q^k$ such distinct polynomials (pick any of the $k$ points and interpolate). Therefore, we will definitely require that $t > k$ if we would like to efficiently list-decode.

### 6.2.1 A Geometric Perspective

For the purpose of this algorithm, we will want to describe the $n$ given points using low-degree planar curves that pass through them; that is, we consider curves $\{(x, y) : Q(x, y) = 0\}$ where $Q(x, y)$ is a low-degree polynomial. Note that we are not restricted to curves with degree one in $y$; in particular, we may describe points on a circle centered at $(0, 0)$ with the equation $x^2 + y^2 - 1 = 0$. Other examples of point sets that may be described using low-dimensional curves are lines, and unions of lines and circles.



In the example on the left, we have a set of 13 points that lie on the union of a circle and two lines. Suppose the point in the center is $(0, 0)$. Then, the set of points lie on the curve described by: $(x^2 + y^2 - 1)(x - y)(x + y) = 0$.

### 6.2.2 A Simple List-Decoding Algorithm

For the list-decoding problem, the intuition is that if $p$ is a polynomial with large agreement, then the curve $y - p(x) = 0$ passes through many of the given points. Therefore, what we will do in the list-decoding algorithm is to first find a low-degree polynomial $Q$ passing through all of the given points, and then show that all low-degree curves that pass through many of the given points divides $Q$. This reduces the list-decoding problem to factorizing a bivariate polynomial over a finite field, for which efficient algorithms do exist.

**Algorithm** List-Decode-RS
**Given:** $n$ distinct points $(x_1, y_1), \ldots, (x_n, y_n)$ in $\mathcal{F}_q^2$.

1. Find $Q(x, y)$ such that
   - $Q$ has low degree: $d_x - 1$ in $x$, $d_y - 1$ in $y$
   - $Q(x_i, y_i) = 0$ for all $i = 1, 2, \ldots, n$
   - $Q \not\equiv 0$.
2. Factor $Q(x, y)$. For every factor of the form $y - p(x)$, if $p$ is a feasible solution, output $p$.

There are two problems that we need to address:

1. Does there exist a low-degree polynomial $Q$ that pass through all the given points, and if so, how can we find one efficiently?
2. Must every low-degree polynomial that pass through many of the given points divide $Q$? For instance, taking $t = 3$ for concreteness; it seems conceivable that we have a polynomial $R(x, y)$ quadratic in $y$ that passes through 6 of the given points that lie on $y - p_1(x), y - p_2(x)$ for two low-degree polynomials $p_1, p_2$, and that $R(x, y)$ divides $Q$, but neither $y - p_1(x)$ nor $y - p_2(x)$ does.

### 6.2.3 Finding $Q$

First, we address the problem of finding $Q$. We may write

$$Q(x,y) = \sum_{i=0,\ldots,d_x-1; j=0,\ldots,d_y-1} c_{ij} x^i y^j$$

Now, the problem reduces to finding the $d_x d_y$ coefficients of $Q$: $c_{ij}, i = 0, 1, \ldots, d_x - 1; j = 0, \ldots, d_y - 1$. Observe that the requirement $Q(x_i, y_i) = 0$ is equivalent to a system of linear constraints on the coefficients $\{c_{ij}\}$. Furthermore, this is a homogeneous system, so it will always have the all 0's solution, corresponding to $Q \equiv 0$. On the other hand, if $d_x d_y > n$, that is, the number of variables is more than the number of linear constraints, then we can always efficiently find a non-zero solution to the linear system that yields a non-zero $Q$ that passing through all the $n$ points.

### 6.2.4 Proof of Correctness

Next, we will have to show that every polynomial $p$ with large agreement with the points $(x_1, y_1), \ldots, (x_n, y_n)$ is a factor of $Q$. More precisely, we are told that:

1. $p(x)$ is a degree $k - 1$ polynomial such that $y - p(x)$ is zero in at least $t$ of the points.
2. $Q(x, y)$ has degree $d_x - 1$ in $x$ and $d_y - 1$ in $y$ and passes through all the points (that is, $Q(x_i, y_i) = 0$ for $i = 1, 2, \ldots, n$).
3. There are $\geq t$ points $(x_i, y_i)$ such that $Q(x_i, y_i) = y_i - p(x_i) = 0$.

For simplicity, we can rewrite these conditions assuming that we are choosing $n$ points on the curve $Q(x, y) = 0$, which yields the following statement:

**Proposition 1** *Suppose that*

1. *$Q(x, y)$ is bivariate polynomial in $x, y$ with degree $d_x - 1$ in $x$ and $d_y - 1$ in $y$.*
2. *$p(x)$ is a degree $k - 1$ polynomial in $x$.*
3. *There are $\geq t$ points $(x_i, y_i)$ such that $Q(x_i, y_i) = y_i - p(x_i) = 0$.*
4. *$t > (d_x - 1) + (k - 1)(d_y - 1)$.*

*Then $y - p(x)$ divides $Q(x, y)$.*

This proposition is a special case of Bezout's Theorem, that says that any two curves that share lots of points in common must share a common factor. Here, $y - p(x)$ is irreducible (over polynomials in $y$ with coefficients from $\mathcal{F}_q[x]$), so it divides $Q(x, y)$. A simple proof of this special case is shown below.

It is also important to note that we only require that the points $(x_1, y_1), \ldots, (x_n, y_n)$ be distinct, and not that $x_1, \ldots, x_n$ be distinct, as in the case for list-decoding Reed-Solomon codes. This allows the list-decoding procedure to be used in a more general setting, as we shall see later.

Proof: View $Q$ is a univariate polynomial in $y$ whose coefficients are univariate polynomials in $x$:

$$q(y) = q_0(x) + yq_1(x) + \ldots + y^{d_y-1}q_{d_y-1}(x)$$

Recall the Factor Theorem for polynomials: $\beta$ is such that $q(\beta) = 0$ iff $y - \beta$ divides $q(y)$. This tells us that $p(x)$ is such that $q(p(x)) \equiv 0$ iff $y - p(x)$ divides $Q(x, y)$. Therefore, to show $y - p(x)$ divides $Q(x, y)$, it suffices to show that $Q(x, p(x))$ is the zero polynomial.

From condition 3, we know that $Q(x_i, p(x_i)) = 0$ for at least $t$ distinct values of the $x_i$'s. On the other hand, $Q(x, p(x))$ as a univariate polynomial in $x$ can be written as:

$$Q(x, p(x)) = q_0(x) + p(x)q_1(x) + \ldots + p(x)^{d_y-1}q_{d_y-1}(x)$$

and has degree at most $(d_x - 1) + (k - 1)(d_y - 1)$. Therefore, if $t > (d_x - 1) + (k - 1)(d_y - 1)$, then $Q(x, p(x)) \equiv 0$ and $y - p(x)$ divides $Q(x, y)$. $\square$

### 6.2.5 Fixing the Parameters

We are now ready to fix the parameters $d_x, d_y$. Recall that we require that:

1. $d_x d_y > n$, so that we have sufficient variables in the linear system for finding $Q$;
2. $t > d_x + kd_y$, to ensure that every polynomial with large agreement is a factor of $Q$.

We want to maximize $t$ under both constraints, and that is optimized by setting $d_x = \sqrt{kn}$ and $d_y = \sqrt{n/k}$, so $d_x + kd_y = 2\sqrt{kn}$. As a polynomial in $y$, $Q$ has degree $d_y$ and therefore at most $d_y$ factors. Hence, there are at most $d_y = \sqrt{n/k}$ polynomials in the list. This yields the following results:

**Theorem 2** *Given a list of $n$ points $(x_1, y_1), \ldots, (x_n, y_n)$ in $\mathcal{F}_q^2$, we can efficiently find a list of all polynomials $p(x)$ of degree at most $k - 1$ that pass through at least $t$ of these $n$ points, as long as $t > 2\sqrt{nk}$. Furthermore, the list has size at most $\sqrt{n/k}$.*

**Theorem 3** *For every $\epsilon > 0$, and for all sufficiently large $n$, there exist:*

1. *A $[n, \epsilon n, (1 - \epsilon)n]_n$ Reed-Solomon code, such that we can efficiently list-decode from agreement in at least $2\sqrt{\epsilon}n$ locations, and size of the list is at most $\sqrt{1/\epsilon}$.*
2. *A $[n, \epsilon^2 n/4, (1 - \epsilon^2/4)n]_n$ Reed-Solomon code such that we can efficiently list-decode from agreement in at least $\epsilon n$ locations, and the size of the list is at most $2/\epsilon$.*

### 6.2.6 Increasing the List-Decoding Radius

Observe that in the proof of correctness, we only require that $Q(x, p(x))$ has degree less than $t$ in $x$. Therefore, it suffices that for all monomials $x^i y^j$ in $Q(x, y)$, we have $i + kj < t$ (instead of the more restrictive constraint that $i < t/2$ and $j < t/2k$). This means that we may consider any $Q(x, y)$ of the form:

$$Q(x, y) = \sum_{i+kj<t} c_{ij} x^i y^j$$

Therefore, the number of coefficients (and thus the number of variables in the linear system) is given by:

$$| \{(i,j) : i + kj < t\} | = \overbrace{t + (t-k) + (t-2k) + \ldots + (t - \frac{t}{k} \cdot k)}^{t/k} = \frac{t}{k} \cdot \frac{1}{2}(t+0) = \frac{t^2}{2k}$$

(instead of $t/2 \cdot t/2k = \frac{t^2}{4k}$ if we consider only $i < t/2$ and $j < t/2k$.) To ensure that the linear system $\{Q(x_i, y_i) = 0 \mid i = 1, 2, \ldots, n\}$ is under-determined, we need $\frac{t^2}{2k} > n$, or equivalently, $t > \sqrt{2kn}$. For such $t$, it suffices to consider $Q$ of the form:

$$Q(x,y) = \sum_{i+kj<t \,|\, j \le \sqrt{2n/k}} c_{ij} x^i y^j$$

This allows us to place an upper bound of $\sqrt{2n/k}$ on the size of list (instead of the crude bound $t/k$).

**Theorem 4** *Given a list of $n$ points $(x_1, y_1), \ldots, (x_n, y_n)$ in $\mathcal{F}_q^2$, we can efficiently find a list of all polynomials $p(x)$ of degree at most $k-1$ that pass through at least $t$ of these $n$ points, as long as $t > \sqrt{2nk}$. Furthermore, the list has size at most $\sqrt{2n/k}$.*

## 6.3 List-decoding concatenated codes (RS with Hadamard)

Recall that the Hadamard code is a $[2^k, k, \frac{1}{2} \cdot 2^k]_2$ code with an encoding function $C : \{0,1\}^k \to \{0,1\}^{2^k}$ that sends a message $c_1, c_2, \ldots, c_k$ to $\{c_1 a_1 + \cdots + c_k a_k\}_{a \in \{0,1\}^k}$. For our analysis of the list-decoding algorithm, we will need the following Johnson-type bound for the Hadamard code, which follows from a simple application of Fourier analysis:

**Theorem 5** *For every $y \in \{0,1\}^n$ and for every $\epsilon > 0$, there are at most $\frac{1}{4\epsilon^2}$ codewords of the $[n, \log n, n/2]_2$ Hadamard code at distance $\le \left(\frac{1}{2} + \epsilon\right) n$ from $y$.*

In addition, we will focus on the code obtained by concatenating the Reed-Solomon Code with the Hadamard code with the following parameters:

| | |
|---|---|
| Reed-Solomon: | $[n, k, n-k]_n$ |
| Hadamard: | $[n, \log n, \frac{1}{2}n]_2$ |
| concatenated: | $[n^2, k \log n, \frac{1}{2}n(n-k)]_2$ |

### 6.3.1 A Simple List-Decoding Algorithm

Consider the naive list-decoding algorithm for the concatenated code from $\left(\frac{1}{2} - \epsilon\right) n^2$ errors, wherein we first list-decode the outer code, and then list-decode the inner code:

**Algorithm** LIST-DECODE-RS-HADAMARD
**Given:** $(z_1, \ldots, z_n) \in \{0,1\}^{n^2}$, where $z_i \in \{0,1\}^n$ is a block corresponding to a codeword of the inner code.

1. Decode $n$ codewords of the outer code assuming $\leq \left(\frac{1}{2} - \frac{\epsilon}{2}\right) n$ errors via a brute force search. That is, for each $i = 1, 2, \ldots, n$, we output a list $L_i$ of all the codewords of the outer code that agree with $z_i$ in at least $\left(\frac{1}{2} + \frac{\epsilon}{2}\right) n$ positions. Note that $|L_i| \leq 1/\epsilon^2$.

2. For $j = 1, 2, \ldots, 1/\epsilon^2$: pick the $j$th element $y_i$ of each list $L_i$, $i = 1, 2, \ldots, n$, and run List-Decode-RS on $(y_1, \ldots, y_n)$ to find all degree $k-1$ polynomials that agree with $(y_1, \ldots, y_n)$ on at least $\frac{1}{2}\epsilon^3 n$ locations.

3. Output the messages corresponding to all such polynomials.

Note that for a random block $z_i$ chosen from $z_1, \ldots, z_n$, $z_i$ agrees with the correct encoding in $\geq \left(\frac{1}{2} + \epsilon\right) n$ locations. Then by Markov's inequality,

$$\Pr_i[z_i \text{ has agreement at least } \left(\tfrac{1}{2} + \tfrac{\epsilon}{2}\right) n \text{ with the correct encoding}] \geq \frac{\epsilon}{2}.$$

Otherwise, the agreement between $(z_1, \ldots, z_n)$ and the correct encoding is $< \frac{\epsilon}{2} \cdot n + 1 \cdot (\frac{1}{2} + \frac{\epsilon}{2})n = (\frac{1}{2} + \epsilon)n$, a contradiction. Hence, at least $\frac{\epsilon n}{2}$ of the lists $L_i$ contains the correct outer codeword. If we pick a random element from each of these lists $L_i$, then we expect an agreement $\geq \frac{\epsilon n}{2} \cdot 1/(1/\epsilon^2) = \frac{\epsilon^3 n}{2}$ between the $n$ field elements we have chosen, and $n$ elements corresponding to the correct Reed-Solomon encoding. By linearity of expectations, this is still true if we fix a random $j \in \{1, 2, \ldots, 1/\epsilon^2\}$ and pick the $j$th element from each of the lists $L_i$. Therefore, if we enumerate over all possible choices of $j$, there is at least one $j$ for which we have agreement $\geq \frac{\epsilon^3 n}{2}$. In this case, List-Decode-RS will succeed as long as:

$$\epsilon^3 n/2 > \sqrt{2kn}, \text{ that is, } k \leq \epsilon^6 n/8$$

For each $j$, we have a list of size $\sqrt{2n/k} = O(1/\epsilon^3)$, so the total size of the list is $O(1/\epsilon^5)$.

**Theorem 6** *For every $\epsilon > 0$, there exists a binary code that is obtained by concatenating a Reed-Solomon Code with a Hadamard Code and that has the following properties:*

1. *It is a $[n^2, \frac{1}{8}\epsilon^6 n \log n, \frac{1}{2}n^2(1 - \epsilon^6/8)]_2$ code.*

2. *It can be efficiently list-decoded from $\frac{1}{2} - \epsilon$ fraction of errors, and the size of the list is $O(1/\epsilon^5)$.*

Finally, observe that we are not restricted to using the Hadamard code as the inner code, as List-Decode-RS-Hadamard will run efficiently as long as the message length of the inner code is logarithmic in the size of the input, as that would already allow us to enumerate all codewords of the inner code in time polynomial in the size of the input and therefore efficiently list-decode the inner code.

### 6.3.2 An Improved List-Decoding Algorithm

We can improve upon the parameters of List-Decode-RS-Hadamard with the simple observation that List-Decode-RS works as long as the input points are distinct; in particular, the evaluations of the polynomial given to us do not have to be at pairwise distinct points. As such, we can concatenate the lists $L_1, \ldots, L_n$, and pass all $|L_1| + \ldots + |L_n| \leq \frac{n}{\epsilon^2}$

points to List-Decode-RS, with a guaranteed agreement of at least $\frac{\epsilon n}{2}$. In this case, List-Decode-RS will succeed as long as:

$$\epsilon n/2 > \sqrt{2k(n/\epsilon^2)}, \text{ that is, } k \leq \epsilon^4 n/8$$

This yields a list of size $\sqrt{2(n/\epsilon^2)/k} = O(1/\epsilon^3)$.

**Algorithm** List-Decode-RS-Hadamard-2
**Given:** $(z_1, \ldots, z_n) \in \{0,1\}^{n^2}$, where $z_i \in \{0,1\}^n$ is a block corresponding to a codeword of the inner code.

1. Decode $n$ codewords of the outer code assuming $\leq \left(\frac{1}{2} - \frac{\epsilon}{2}\right) n$ errors via a brute force search to obtain $n$ lists $L_1, \ldots, L_n$.
2. Run List-Decode-RS on $L = L_1 \cup \ldots \cup L_n$ to find all degree $k-1$ polynomials that agree with $L$ on at least $\frac{1}{2}\epsilon n$ locations.
3. Output the messages corresponding to all such polynomials.

**Theorem 7** *For every $\epsilon > 0$, there exists a binary code that is obtained by concatenating a Reed-Solomon Code with a Hadamard Code and that has the following properties:*

1. *It is a $[n^2, \frac{1}{8}\epsilon^4 n \log n, \frac{1}{2}n^2(1 - \epsilon^4/8)]_2$ code.*
2. *It can be efficiently list-decoded from $\frac{1}{2} - \epsilon$ fraction of errors, and the size of the list is $O(1/\epsilon^3)$.*