6.897 Algorithmic Introduction to Coding Theory	October 29, 2001
Lecture 12	
Lecturer: Madhu Sudan	*Scribe: George Savvides*

Today's topics:

1. Combinatorics (revisited), including two new proofs of the Johnson bound.
2. List decoding for Reed-Solomon codes.

1 Combinatorics revisited

Let us recall the notion of an (e, ℓ)-error-correcting code [1]

Definition 1 *Given an* $[n, k, d]_q$*-code* C*, we say that* C *is an* (e, ℓ)-error correcting code*, if for* any *received vector* $\mathbf{r} \in \mathbb{F}_q^n$*, there exist at most* ℓ *codewords* $\mathbf{c}_1 \ldots, \mathbf{c}_\ell \in C$ *such that* $\Delta(\mathbf{c}_i, \mathbf{r}) \leq e$*.*

In other words the code C can correct up to e errors with lists of size ℓ. The natural algorithmic problem arising out of the above definition is whether (and how) we can find all such codewords in polynomial time. That is:

Given: $\mathbf{r} \in \mathbb{F}_q^n$, an arbitrary received vector).

Output: A set (list) of codewords $\{c_1, \ldots, c_\ell\}$ such that every codeword $\mathbf{c}$ that satisfies $\Delta(\mathbf{c}, \mathbf{r}) \leq e$ is included in this set.

We will require that our algorithm's running time be polynomial in the *input* size. Notice that it is possible that if ℓ is superpolynomial, then such an algorithm can not exist! So, a prerequisite for the existence of this algorithm is that the code be an (e, ℓ)-error-correcting code for ℓ that is a polynomial in the block length of the code. When do we know that an $[n, k, d]_q$ code is also an (e, ℓ)-error-correcting code for some ℓ that is polynomial in n? The Johnson bound ought to come to mind at this stage. (Though, given the general sentiment against naming bounds by discoverer's names, one can be forgiven for not coming up with the right name. Hopefully the question still does ring a bell.) We recall the Johnson bound below:

Theorem 2 (Johnson bound) *Let* $\epsilon = \frac{e}{n}$ *and* $\delta = \frac{d}{n}$*. An* $[n, k, d]_q$*-code is an* (e, n)*-error-correcting*[2] *code provided that*

$$\epsilon \leq \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\delta}\right)$$

Recall that in class we proved the binary case of the theorem, where the condition on ϵ was

$$\epsilon \leq \frac{1}{2}\left(1 - \sqrt{1 - 2\delta}\right).$$

[1] Note that in previous lectures, we used to call this a (t, ℓ)-error-correcting code. We are changing the parameter label for this lecture, since we end up using t for the "opposite" of e, i.e., the number of agreements rather than number of errors.

[2] yes, n is both the size of the codewords and the size of the list

The Johnson bound applied to Reed-Solomon codes In the case of RS codes, $q \to \infty$, so we can simplify the bound above. In particular, we get that the $[n, k, d]_q$ RS-code is also an $(\epsilon n, n)$-error-correcting code for $\epsilon \leq 1 - \sqrt{1-\delta}$. Using the fact that the rate R of such a code is $1 - \delta$, the above bound says that the "list-of-n decoding radius" of the code is $1 - \sqrt{R}$. To compare this with the list-of-1 decoding radius (or the unambiguous decoding radius), in such case we can only allow $\epsilon \leq \frac{1-R}{2}$. Thus, as we would hope, the list-decoding radius is strictly larger than the unambiguous decoding radius for any Reed-Solomon code of rate less than 1.

Furthermore, to highlight the difference, let us consider an extremal case when we let the rate approach zero:

1. $\lim_{R \to 0} 1 - \sqrt{R} = 1 \quad (\text{so } \epsilon \to 1)$

2. $\lim_{R \to 0} \dfrac{1-R}{2} = 1/2 \quad (\text{so } \epsilon \to 1/2)$

Thus in Case 2 (unambiguous decoding), a necessary condition for successful decoding is that the fraction of errors be bounded from above by the fraction of correct information. However, list-decoding seems feasible (combinatorially, at least) even when this condition is violated — i.e., when the amount of correct information is overwhelmed by the amount of erroneous information. As we will see over the course of the next couple of lectures, this barrier can be overcome even algorithmically! But first, we will celebrate the Johnson bound (er ... yes, we still don't have a better way to name the bounds) by giving two alternate proofs of the bound.

2 Two proofs of the Johnson bound

The elegance and beauty of some of the (numerous) proofs of the Johnson bound are part of the reason why this particular bound is of interest to us. Below we will see two new proofs of the Johnson bound for the case of Reed-Solomon codes. Let us first recall the problem being considered:

> Given n distinct points $x_1, \ldots, x_n \in \mathbb{F}_q$ and n values $r_1, \ldots .r_n \in \mathbb{F}_q$, bound the number of polynomials p of degree at most k, such that $p(x_i) = r_i$ for at least t values of $i \in [n]$.

Note that we have shifted the focus to the number of agreements rather than the number of errors. This is a better choice, since this is the smaller number and hence it is a more refined number to look at (as opposed to the number of disagreements). Our bounds will be of the form "the number of such polynomials is small, provided $t \geq c\sqrt{kn}$. We now give the proofs.

2.1 Using the Inclusion-Exclusion principle

The inclusion-exclusion principle shows how to count the number of elements in the union of ℓ sets, given the sizes of all interesections. While the full formula (not given here) gives the size exactly, portions of the formula are known to give bounds on the size of the union. The union bound is a simple version which says that the size of the union is *at most* the sum of the sizes of the individual sets. Looking at sizes of pairwise intersections gives the lower bound below:

Proposition 3 (Weak Inclusion-Exclusion Principle) *For any ℓ subsets $S_1, \dots, S_\ell$ of some finite universe X, the following inequality holds:*

$$\left|\bigcup_{j=1}^{\ell} S_j\right| \quad \geq \quad \sum_{j=1}^{l} |S_j| \quad - \quad \sum_{1 \leq j_1 < j_2 \leq l} \left|S_{j_1} \bigcap S_{j_2}\right|.$$

Exercise: Prove Proposition 3.

Now, we show how to apply the Inclusion-Exclusion Principle to our problem. Suppose $p_1, \dots, p_\ell$ are degree k polynomials that agree with the given points on at least t places. We define S_j, $j \in [\ell]$ as follows:

$$S_j = \{i \in [n] \mid p_j(x_i) = r_i\}.$$

We note that S_j's satisfy the following properties:

- For every $j \in [n]$, $|S_j| \geq t$: By hypothesis.
- $\left|\bigcup_{j=1}^{\ell} S_j\right| \leq n$: Obvious, since each $S_j \subseteq [n]$.
- For every distinct j_1, j_2, $|S_{j_1} \bigcap S_{j_2}| \leq k$: Since $p_{j_1}(x_i) = p_{j_2}(x_i)$ for every $i \in S_{j_1} \bigcap S_{j_2}$, and two distinct degree k polynomials may agree on at most k distinct elements of $\mathbb{F}_q$.

From the above, and the inclusion-exclusion principle, and the fact that there are $\binom{\ell}{2}$ pairs of intersections to consider, we have:

$$n \geq \ell \cdot t - \binom{\ell}{2} \cdot k. \tag{1}$$

Further, by monotonicity, we actually have

$$\forall \ell' \leq \ell \quad n \geq \ell' \cdot t - \binom{\ell'}{2} \cdot k. \tag{2}$$

(If there are ℓ polynomials with agreement of t, then are also $\ell' \leq \ell$ polynomials with agreement t.)

The following unremarkable and tedious lemma finishes off the proof:

Lemma 4 *Suppose n, k, t are positive integers such that $t > \sqrt{2kn}$. Then $n < \ell' \cdot t - \binom{\ell'}{2} \cdot k$ for $\ell' = \lceil \frac{2n}{t} \rceil$.*

Proof Let $g(x) = \frac{k}{2}x^2 - \left(t + \frac{k}{2}\right)x + n$. Note that $\frac{2n}{t} < \ell' \leq \frac{2n}{t} + 1$. By convexity of $g(\cdot)$ we have $g(\ell') \leq \max\left\{g\left(\frac{2n}{t}\right), g\left(\frac{2n}{t} + 1\right)\right\}$. We argue that both $g\left(\frac{2n}{t} - 1\right)$ and $g\left(\frac{2n}{t}\right)$ are negative. First,

$$\begin{aligned} g(\frac{2n}{t}) &= \frac{2kn^2}{t^2} - \frac{n(2t+k)}{t} + n \\ &= \frac{n(2kn - t^2 - kt)}{t^2} \\ &< \frac{-nkt}{t^2} \quad (\text{Using } 2nk < t^2) \\ &\leq 0. \end{aligned}$$

Next, we have:

$$
\begin{aligned}
g(\frac{2n}{t}+1) &= \frac{k(2n+t)^2}{2t^2} - \frac{(2t+k)(2n+t)}{2t} + n \\
&= \frac{4n^2k + 4nkt + t^2k - 4nt^2 - 2t^3 - 2nkt - kt^2 + 2nt^2}{2t^2} \\
&= \frac{4n^2k + 2nkt - 2nt^2 - 2t^3}{2t^2} \\
&< \frac{t^2(2n+t) - 2nt^2 - 2t^3}{2t^2} \quad \text{(Using } 2nk < t^2\text{)} \\
&= \frac{(2n+t) - 2n - 2t}{2} \\
&= \frac{-t}{2} \\
&< 0.
\end{aligned}
$$

■

We conclude with the first bound.

Theorem 5 *Given points $\{(x_i, r_i)\}_{i=1}^n$ there are at most $\frac{2n}{t}$ polynomials p of degree at most k that satisfy $|\{i \in [n] \mid p(x_i) = y_i\}| \geq t$ if $t > \sqrt{2kn}$.*

Note that this bound as well as the next one are just weak versions of the bound given in Theorem 2. Both apply generally to codes. However we are just specializing the arguments to the case of polynomials, to keep the notation simple. We just repeat them here to give further intuition into what they are trying to prove, and to illustrate the simple proof techniques.

2.2 Using graph theory

Our second proof of the "Johnson-like bound" goes via graph theory. We model the problem graph-theoretically as follows:

Given a received vector $\mathbf{r}$ and codewords $\mathbf{c}_1, \ldots, \mathbf{c}_\ell$ we construct a bipartite graph B with vertex set $L \cup R$ and edges being a subset of $L \times R$ as follows:

Left vertices: $L = [n]$, corresponding to the n coordinates of the received vector.

Right vertices: $R = [\ell]$ corresponding to the ℓ codewords $\mathbf{c}_1, \ldots, \mathbf{c}_\ell$.

Edges: There is an edge between $i \in L$ and $j \in R$ iff $r_i = (\mathbf{c}_j)_i$.

The properties of the problem at hand translate to graph theoretic properties as follows:

Distance property: Two codewords can't agree with each other on k indices. In particular this implies that two coordinates can't agree with $\mathbf{r}$ on the same subset of k indices. Thus the graph B does not contain $K_{k,2}$, the complete bipartite graph with k vertices on the left and 2 vertices on the right, as a subgraph.

Agreement property: The agreement between codewords and $\mathbf{r}$ implies that every vertex on the right is adjacent to at least t vertices on the left. For simplicity in the analysis we will throw away some edges in the graph, so as to get right vertices to have degree exactly t. Notice the graph still does not have any $K_{k,2}$ after this deletion.

We will see that these properties will allow us to conclude that the graph doesn't have too many vertices on the right (so ℓ is "small", as desired). Such bounds are called "Zarankiewicz bounds" after a Polish mathematician Kazimierz Zarankiewicz who studied such forbidden subgraph problems.

Lemma 6 *If a bipartite graph $B = (L, R, E)$, with $|L| = n$ and $|R| = \ell$, has right degree t and no $K_{k,2}$,*

$$\ell < \frac{n(t-k)}{t^2 - kn} \quad \textit{provided } t^2 > nk.$$

Proof We prove the lemma using a probabilistic argument. Let d_i denote the degree of the ith vertex in L. Note that $\sum_i d_i = \ell \cdot t$.

Let p_i denote the probability, when two distinct elements j_1 and j_2 are picked uniformly at random from R, that both j_i and j_2 are adjacent to i. We have:

$$p_i = \frac{\binom{d_i}{2}}{\binom{\ell}{2}} = \frac{d_i^2 - d_i}{\ell^2 - \ell}.$$

Now consider the expected number of common neighbours that distinct elements j_1 and j_2 have when they are picked uniformly at random from R. This quantity is given by

$$\sum_i p_i = \frac{1}{\ell^2 - \ell} \sum_{i=1}^{n} (d_i^2 - d_i).$$

Since this quantity better be less than k, we get

$$\frac{1}{\ell^2 - \ell} \sum_{i=1}^{n} (d_i^2 - d_i) < k.$$

The non-trivial component in the inequality above is $\sum_{d_i^2}$. Subject to the condition that $\sum_i d_i = \ell t$, this quantity is minimized when all the d_i's are equal This is proves via a standard inequality known as the Cauchy-Schwartz inequality. For any two real vectors $\mathbf{a} = \langle a_1, \ldots, a_n \rangle$ and $\mathbf{b} = \langle b_1, \ldots, b_n \rangle$, the Cauchy-Schwartz inequality says that $\langle \mathbf{a}, \mathbf{b} \rangle^2 \leq (\|\mathbf{a}\|^2 \cdot \|\mathbf{b}\|^2)$. Applying it to $\mathbf{a} = \langle d_1, \ldots, d_n \rangle$ and $\mathbf{b} = \langle 1, \ldots, 1 \rangle$, we get $(\sum_{i=1}^n d_i)^2 \leq n \sum_{i=1}^n d_i^2$. Thus the inequality above becomes:

$$\frac{1}{n} \left(\sum_{i=1}^{n} d_i \right)^2 - \sum_i d_i < k(\ell^2 - \ell).$$

$$\Leftrightarrow \frac{\ell^2 t^2}{n} - \ell t < k(\ell^2 - \ell).$$

$$\Leftrightarrow \ell \left(\frac{t^2}{n} - k \right) < t - k.$$

Thus we have

$$\ell < \frac{n(t-k)}{t^2 - kn} \quad \text{provided } t^2 > nk.$$

■

Theorem 7 *Given points $\{(x_i, r_i)\}_{i=1}^n$ there are at most $\frac{n(t-k)}{t^2-kn}$ polynomials p of degree at most k that satisfy $|\{i \in [n] \mid p(x_i) = y_i\}| \geq t$ if $t > \sqrt{kn}$.*

3 List decoding for RS codes

We now move on to the algorithmic versions of Theorems 5 and 7. We state the problem next.

> **List decoding of Reed-Solomon codes:**
> GIVEN: Point pairs $\{(x_i, y_i) \mid i \in [n]\}$ with each pair being distinct and parameters k and t.
> FIND: A list of all polynomials p of degree at most k that satisfy $p(x_i) = y_i$ for at least t values of $i \in [n]$.

Our goal is to get a solution that works provided $t > \sqrt{kn}$ (so as to match Theorem 7). Today we will settle for any $t = \Theta(\sqrt{kn})$. The algorithm is actually pretty simple. We motivate it by looking at a toy problem.

3.1 A toy problem

Suppose the message polynomial is a polynomial p_1 and the errors in our point pairs actually come from a second polynomial p_2. It is clear that our algorithm should output as answers p_1 and p_2 (assuming both polynomials are represented often enough). How do we come up with an algorithm whose output is "The answer is $p_1(\cdot)$ OR $p_2(\cdot)$"? The Boolean OR in the statement seems to be hard to capture algebraically. We attempt to capture this by introducing *two* variables x and y representing the two coordinates of the pairs that are given to us and noticing that it is easy to come up with an algebraic *relation* explaining all the given points.

The given set of points satisfy:

$$(y_i - p_1(x_i)) \cdot (y_i - p_2(x_i)) = 0.$$

(In other words, multiplication captures the OR of two relations). Indeed if we can find two polynomials p_1 and p_2 such that the above relation is satisfied by all pairs of points, then we could output p_1 and p_2 as our solution, and this would seems to be good enough. (We will prove later that this would suffice, but now we turn to the more important question.) The question is, how do we find such polynomials?

To find p_1 and p_2, we expand the quadratic expression above, and see that there must exist two polynomials $B(x)$ and $C(x)$ such that the following conditions hold:

1. For every $i \in [n]$, $y_i^2 - B(x_i)y_i + C(x_i) = 0$.
2. There exist polynomials p_1 and p_2 such that
   - $B(x) = p_1(x) + p_2(x)$.
   - $C(x) = p_1(x) \cdot p_2(x)$.
3. $\deg(B) \leq k$ and $\deg(C) \leq 2k$.

As in previous cases (e.g., the Welch-Berlekamp algorithm), we ignore conditions that seem to be hard to enforce and hope the rest suffice to find the solution anyway. In the above set of conditions, (1) and (3) are easy to enforce (we will see so shortly), but condition (2) is not! So we will omit condition (2) and look for B and C satisfying the remaining conditions. This motivates the following algorithm.

Algorithm for toy problem:

Step 1: find B and C such that $\forall\ i \in [n]$:

- $y_i^2 - B(x_i)y_i + C(x_i) = 0$,
- $\deg(B) \leq k$, and
- $\deg(C) \leq 2k$,

provided such a pair exists.

Step 2: Factor the polynomial $y^2 - B(x)y + C(x)$. If it factors into $(y - f(x)) \cdot (y - g(x))$, output f and g.

As in the case of the Welch-Berlekamp algorithm, we have to argue that this algorithm is *efficient* and *correct* and we go through the usual steps to do so. Btw, this paradigm is not special to coding theory, but rather to algebraic algorithms, where one finds out enough information about the solution to outline an algorithm, and then shows that the information is sufficient to have pinned down the answer separately.

We start by describing the algorithmic complexity of the two steps above:

Step 1: We can approach this problem through linear algebra. We wish to find polynomials B and C, or rather their coefficients so that the constraints of Step 1 are satisfied. The constraints are linear in the coefficients (though not linear in x_i's and y_i's) and so if a solution exists, one can be found efficiently.

Step 2: This specific case is easy to handle based on the standard formula for computing roots of quadratic equations. However, we will appeal to a more general solution. It turns out that the task of factoring multivariate polynomials is solvable very efficiently in general. Three indepent works in the eighties, Grigoriev [7], Kaltofen [8], and Lenstra [10], addressed this problem. At the very least, they show that every polynomial in a constant number of variables can be factored in time polynomial in the degree of the input polynomial. This is sufficient for our purposes. But in case you care, Kaltofen and Trager [9] give much stronger results, essentially giving running times that are polynomial in the degree of the polynomials *and* the number of variables, provided you let them pick the representation of the polynomials. A note on the dependence on the field size. If the field is finite and has characteristic p (i.e., the field is an extension of $\mathbb{F}_p$ for some prime p) then the running time is polynomial in p and the logarithm of the field size. It can also be made purely polynomial in the logarithm of the field size, if we allow the algorithm to be randomized. All of this is inherited from the underlying algorithms for factoring univariate polynomials (see [2, 3, 4] etc.). Over the rationals, the algorithms run deterministically in time polynomial in the input size, again inheriting their performance from the univariate case, solved by Lenstra, Lenstra, and Lovasz [11].

Back to the algorithm at hand, notice that Step 1 requires that a solution exist, and Step 2 has a solution only if a possible factorization exists. Below we argue that the solutions do exist assuming our problem instance is fitting the promises of the toy problem.

3.2 Correctness of algorithm for the toy problem

We argue the correctness of the two steps in order.

Claim 8 *Under the hypothesis of the toy problem, a solution to Step 1 exists.*

We already argued this in motivating the algorithm, so we won't restate the proof. We move on to the more important claim, which states that any solution to Step 1, has the "right" factorization. This claim is important in that its proof generalizes naturally to the "non-toy case".

Claim 9 *If B and C are any solutions to Step 1 and p is any degree k polynomial that agrees with the given point set on $t \geq 2k+1$ points, then $y - p(x) \mid y^2 - B(x)y + C(x)$.*

Remark: Recall that the above claim is a specialization of "Bezout's theorem in the plane" — something we alluded to in our coverage of algebraic-geometry codes. This theorem (family of theorems, really) says that two algebraic curves in the plane can not share too many points without sharing a common component.rather Below we find a proof of this fact in this special case. We will generalize the proof a little more, but not completely, by the end of this lecture.

Proof Before we go into the proof of this particular case, let's see how we may prove that some polynomial of the form $y - p(x)$ divides some other polynomial. The classical way to do this, is to think of the polynomial $Q(x,y) = y^2 - B(x)y + C(x)$ as a polynomial $Q(y)$ only in y with coefficients that happen to be from $\mathbb{F}_q[x]$. To prove that some polynomial $y - \alpha$ divides $Q(y)$, we need to show that $Q(\alpha) = 0$. In our case, this says we should consider the quantity $g(x) = Q(x, p(x))$, which happens to be a polynomial in x and show that it is zero. Now the question has turned into one about showing a polynomial in x is zero. The standard method for this step is to bound the degree of g from above, and show that g has more zeroes than its degree. This is exactly what we do below.

Let $g(x) = Q(x, p(x))$. By inspection, we note that $g(x)$ is a polynomial of degree at most $2k$ in x. Now consider an index i such that $y_i = p(x_i)$. For such an i, we have

$$\begin{aligned} g(x_i) &= Q(x_i, p(x_i)) \quad \text{(By definition of } g) \\ &= Q(x_i, y_i)) \quad \text{(By choice of } i) \\ &= 0 \quad \text{(By property of } Q \text{, returned in Step 1)} \end{aligned}$$

Since $y_i = p(x_i)$ on at least $2k+1$ points, we have g has more roots than its degree, and thus $g(x) = 0$. Now, viewing Q as a polynomial in y with coefficients from $\mathbb{F}_q[x]$, we get that $Q(p(x)) = 0$, and so $y - p(x)$ divides $Q(y)$. ■

In our treatment of the "toy problem" above, very little was specific to having the solution come from one of two polynomials. We could easily handle 20 polynomials in the same way. Note however that the answers must come from a *finite* set of polynomials.

General case

We now move to the general case, (i.e., no longer sticking with the toy problem). Surprisingly enough, we won't change the algorithm, and even the analysis will change only mildly!

RS-list-decoder-1:

Given Set of n distinct pairs $\{(x_i, y_i)|i \in [n]\}$, and integers t and k.

Parameters: ℓ and D to be set later.

Step 1: Find a bivariate polynomial $Q(x, y)$ of degree ℓ in y and degree D in x such that:

- $Q(x_i, y_i) = 0 \quad \forall i$.
- $Q \not\equiv 0$.

Step 2: Factor $Q(x, y)$ and output the list of polynomials p of degree at most k provided:

- $y - p(x) \mid Q(x, y)$, and
- $p(x_i) = y_i$ for at least t values of $i \in [n]$.

It is evident that this still works efficiently — Step 1 is still solving a linear system and Step 2 is factoring a multivariate polynomial. What we need to argue is the correctness. The main concern in the correct is that we have no reason to believe a solution should exist for Step 1. It turns out that this is completely trivial — in a technical sense! We will show that really nice polynomials fitting the condition of Step 1 exist without any conditions on the points (only on the degree of Q).

Proposition 10 *Given any set $S = \{(x_i, y_i)|i \in [n]\}$, of n pairs from $\mathbb{F}_q \times \mathbb{F}_q$ there is a non-zero polynomial Q with $\deg_x(Q) \leq \frac{n}{\ell}$ and $\deg_y(Q) \leq \ell$, such that $Q(x_i, y_i) = 0$ for every $i \in [n]$.*

Proof Note that the linear system to be solved, to find Q is a *homogenous* linear system (which is why we go to the trouble of avoiding the all zero solution), and it has $(\ell + 1)(\lfloor n/\ell \rfloor + 1) > n$ coefficients, and only n constraints. So the system is "underdetermined" and thus has a non-trivial solution. ■

As a special case, we could ask for polynomials of degree $\sqrt{n}$ in x and y, and Proposition 10 asserts that such a polynomial fitting n points always exists. Notice that we are gaining upon univariate fits, by a whole $\sqrt{n}$ factor and this is where the crucial advantage comes from. Of course, the proof makes it totally clear that this Q-polynomial is very unlikely to be uniquely specified at this stage — but this is usually tackled by algebraic means.

We will prove a stronger version of Claim 9 below (actually, state a stronger version and let the reader prove it as an exercise).

Lemma 11 *If Q is any solution to Step 1 and p is any degree k polynomial that agrees with the given point set on $t > D + \ell \cdot k$ points, then $y - p(x) \mid Q(x, y)$.*

The proof is totally identical to that of Claim 9, with the minor change that the degree of the g polynomial (and hence the agreement required) go up to $D\ell$. The reader should verify the claim though!

Putting Proposition 10 and Lemma 11 together, we get the following theorem.

Theorem 12 *The algorithm* **RS-list-decoder-1** *with the setting $D = \sqrt{nk}$ and $\ell = \sqrt{n/k}$ solves the* **RS list decoding problem** *in polynomial time provided $t > 2\sqrt{kn}$.*

Proof By Proposition 10, a non-zero polynomial $Q(x,y)$ with $\deg_x(Q) \leq D$ and $\deg_y(Q) \leq D$ does exist. By Lemma 11, such a polynomial will have $y - p(x)$ as a factor, if p is any degree k polynomial that has agreement $D + \ell \cdot k = 2\sqrt{nk}$. ■

This bound is not optimal. Notice that the "Johnson bound" from Theorem 7 only requires $t > \sqrt{kn}$ and so we are off by a factor of 2. We can actually get rid of one $\sqrt{2}$ factor in t by a close examination of the proof, and defining and controlling the degree of Q more carefully. We will see this at the beginning of the next lecture. The second $\sqrt{2}$ factor is more important. We will get rid of this in a much more involved algorithm later next lecture.

Bibliographic Notes

The first proof of the Johnson bound (inclusion-exclusion) is a "folklore" result in the CS literature (cf. [5, 6] etc.). The second proof of the Johnson bound is due to Jaikumar Radhakrishnan [12].

The algorithm for list-decoding of Reed-Solomon codes is from Sudan [13]. Most of the key insights, and in particular the solution to the "toy problem", is from the work of Ar, Lipton, Rubinfeld, and Sudan [1]. The main observation in [13] is the "triviality proposition", Proposition 10.

References

[1] Sigal Ar, Richard Lipton, Ronitt Rubinfeld, and Madhu Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing*, 28(2):488–511, 1999.

[2] Elwyn R. Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46:1853–1859, 1967.

[3] Elwyn R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24:713–735, 1970.

[4] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics **138**, Springer Verlag, Berlin, 1993.

[5] Uriel Feige and Carsten Lund. On the hardness of computing the permanent of random matrices. *Computational Complexity*, 6(2):101–132, 1997.

[6] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *SIAM Journal on Discrete Mathematics*, 13(4):535–570, November 2000.

[7] Dima Grigoriev. Factorization of polynomials over a finite field and the solutions of systems of algebraic equations. *Translated from Zapiski Nauchnykh Seminarov Lenningradskogo Otdeleniya Matematicheskogo Instituta im. V. A. Steklova AN SSSR*, 137:20–79, 1984.

[8] Erich Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM Journal on Computing*, 14(2):469–489, 1985.

[9] Erich Kaltofen and Barry Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *Journal of Symbolic Computation*, 9(3):301–320, 1990.

[10] Arjen K. Lenstra. Factoring multivariate polynomials over finite fields. *Journal of Computer and System Sciences*, 30(2):235–248, April 1985.

[11] Arjen K. Lenstra, Hendrik W. Lenstra, and Lászlo Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[12] Jaikumar Radhakrishnan. Personal communication, January 1997.

[13] Madhu Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.