6.897 Algorithmic Introduction to Coding Theory October 23, 2002

Lecture 13

Lecturer: Madhu Sudan *Scribe: Brian Dean*

Today we discuss algorithms for list decoding of Reed-Solomon codes. These algorithms extend and generalize our approaches for unambiguous decoding of Reed-Solomon codes.

1 The List-Decoding Radius

Recall from our earlier discussion of the Elias bound the notion of an (e, l)-error correcting code: C is an (e, l)-ECC if for every received vector $r \in \Sigma^n$ we have at most l codewords in a Hamming ball of radius e around r. That is, there are at most l codewords whose transimission could have resulted in r being received, given that our channel induces at most e errors.

Every code of minimum distance d is a $(\frac{d-1}{2}, 1)$-ECC. This motivated us to study the unambiguous decoding problem for the case when at most $\frac{d-1}{2}$ errors affect every transmission. If more errors are known to occur, then many codewords may serve as valid decodings for a particular received vector r. How many? One may derive a q-ary version of the Johnson bound which gives the following:

Claim 1 *A code of minimum distance d and block length n is a $(n - \sqrt{n(n-d)}, poly(n))$-ECC.*

To motivate the strength of this bound, consider a code with minimum distance $d = 0.99n$. We will be able to unambiguously decode a received vector as long the number of errors is at most $\frac{d-1}{2}$; that is, as long as e is at most roughly $0.5n$. However, by invoking the above bound we find we can tolerate up to as many as $0.9n$ errors while being able to decode a received vector into a polynomially-sized list of candidate codewords.

Recall that the problem of list decoding involves finding all codewords within Hamming distance e of a received vector r. Hence, as long as $e \leq n - \sqrt{n(n-d)}$, there will be only polynomially many such codewords, which motivates us to search for an algorithm to produce them all in polynomial time.

Consider now the case of Reed-Solomon codes. For such codes, we know that $n - d = k$, where k denotes the degree of a message polynomial. Therefore, we will have a polynomial-size list of candidate codewords as long as e is at most roughly $n - \sqrt{kn}$. Otherwise stated,

Claim 2 *If a transmitted vector and its corresponding received vector agree in at least $\sqrt{kn}$ coordinates, then the received vector decodes to a polynomially-sized list of codewords.*

Although this claim makes no assertions about whether or not efficient list decoding is possible with only $\sqrt{kn}$ agreements, it turns out that this is possible in polynomial time (although a bit complicated). In the rest of this lecture, we'll describe a simpler polynomial-time algorithm for list decoding when the number of agreements is at least $2\sqrt{kn}$.

2 List Decoding of Reed-Solomon Codes

The input to the Reed-Solomon list decoding problem is the following:

- Values $\alpha_1 \dots \alpha_n$ at which the message polynomial is evaluated.
- A received vector $< r_1 \dots r_n >$.
- The degree, k, of the message polynomial.
- The minimum number of agreements, t, between transmitted and received vectors.

Our goal is to find all polynomials p of degree at most k such that $p(\alpha_i) = r_i$ for at least t values of i. These are the polynomials which represent messages that are candidate decodings of r.

2.1 Warmup: A Toy Problem

In order to begin developing intuition about the list decoding problem for RS codes, consider first a simple "toy" problem with two candidate message polynomials of degree at most k: the actual message p, and an "error" polynomial p'. Suppose that for all coordinates r_i of the received vector, either $p(\alpha_i) = r_i$ or $p'(\alpha_i) = r_i$. That is, all errors in transmission can be "explained" by the polynomial p'.

If the number of coordinates of r at which p and p' agree (i.e. $p(\alpha_i) = p'(\alpha_i)$) is small, then we have no way of determining from just the received vector which of p and p' is the "real" message and which is the "error". Both solutions should look equally feasible by symmetry, and our list decoding algorithm should produce both of them.

We can now generalize slightly the notion of the error-locator polynomial from the Welch-Berlekamp algorithm. Such a generalization is needed because our original notion of an error-locator polynomial, a polynomial with zeros at all α_i's which are errors (where $p(\alpha_i) \neq p'(\alpha_i)$), is now too computationally expensive to manage. Since there are now potentially many error points, such a polynomial would need excessively high degree. To overcome this difficulty, we will characterize error points not with an explicit error-locator polynomial, but rather implicitly as follows. Since all coordinates r_i in the received vector satisfy $r_i = p(\alpha_i)$ or $r_i = p'(\alpha_i)$, they all must satisfy

$$(r - p(\alpha))(r - p'(\alpha)) = 0.$$

Expanding this product, we obtain

$$r^2 - a(\alpha)r + b(\alpha) = 0,$$

where $a(x) = p(x) + p'(x)$ and $b(x) = p(x)p'(x)$. This motivates the following decoding algorithm:

1. Find polynomials $a(x)$ of degree at most k and $b(x)$ of degree at most $2k$ such that $r_i^2 - a(\alpha_i)r_i + b(\alpha_i) = 0$ for all $i = 1 \dots n$.
2. Factor $r^2 - a(x)r + b(x)$ into $(r - q(x))(r - q'(x))$.
3. Output q and q'.

Step 1 is straightforward as it involves finding a non-trivial solution to a linear system of equations to obtain the coefficients of a and b. We know a solution exists, since our original p and p' give such a solution. Step 2 is a bit less clear, as this involves computing square roots of polynomials, but it turns out that this can in fact be accomplished in polynomial time, as long as square roots can be efficiently computed in the base field. We omit further details.

The preceding approach works fine as long as our errors have a nice "algebraic description" – in this case in terms of a single polynomial p'. In the next section we proceed to generalize this approach further.

2.2 List Decoding with "Nice Errors"

We now generalize our notion of an error-locator polynomial for the case where errors may not come from a single polynomial p', but in which they still have a simple algebraic description. That is, we may no longer be able to describe all errors as satisfying $y - p'(x) = 0$, but we will rather try to express them as zeros of a multivariate polynomial $E(x, y)$ with the following properties:

- E is not identically zero.
- E has small degree in x and y.
- $E(\alpha_i, r_i) = 0$ for all error coordinates i (for which $p(\alpha_i) \neq r_i$).

For simplicity, we now adopt the following assumption, which we will later show how to remove.

Assumption 3 *There exists an error-locator polynomial $E(x, y)$ of degree at most D satisfying the conditions above, where D is "reasonably small".*

This essentially is assuming that our errors are "nice" in the sense that they have a simple description in terms of a low-degree algebraic curve. Under this assumption, how can we efficiently compute E and p? Well, we begin by noticing that for every $i = 1 \ldots n$, either $E(\alpha_i, r_i) = 0$ or $p(\alpha_i) = r_i$. Let us therefore define $Q(x, y)$ as

$$Q(x, y) = E(x, y)(y - p(x))$$

For all coordinates i, we will have $Q(\alpha_i, r_i) = 0$. The polynomial Q is completely analogous to the expression $(y - p'(x))(y - p(x))$ from the initial toy example; in that case we had $E(x, y) = y - p'(x)$. We can now generalize our earlier decoding algorithm:

1. Find $Q \neq 0$ of small degree such that $Q(\alpha_i, r_i) = 0$ for all $i = 1 \ldots n$. Under our simplifying assumption, we know that such a Q exists with degree at most $D + k$.
2. Factor Q.
3. Output all p such that $y - p(x)$ divides $Q(x, y)$ and $p(\alpha_i) = r_i$ for all $i = 1 \ldots n$.

As before, step 1 involves simply finding a non-trivial solution to a linear system to determine the coefficients of terms of Q. Such a solution is guaranteed to exist due to our simplifying assumption. There may even be many non-trivial solutions for Q, so we must address how this may affect the output. Remarkably, it turns out that even though many different polynomials may be valid solutions for Q, all of these are equally suitable for computing p. That is,

Proposition 4 *If the degree of p is at most k and if the number of coordinates i for which $p(\alpha_i) = r_i$ is at least t, then $y - p(x)$ will divide $Q(x, y)$.*

Proof: Define $g(x) = Q(x, p(x))$. Since Q has degree at most $D + k$ and p has degree at most k, the degree of g will be at most $k(D + k)$. Observe now that by our construction of g, $y - p(x)$ will divide $Q(x, y)$ if and only if g is identically zero. We can proceed to prove that g is identically zero by showing that it has strictly more than $k(D + k)$ zeros (if a polynomial has more zeros than its degree, it must be identically zero). What are obvious candidates for zeros of g? Well, consider all coordinates i for which $p(\alpha_i) = r_i$. For any such coordinate, we will have $g(\alpha_i) = Q(\alpha_i, p(\alpha_i)) = Q(\alpha_i, r_i) = 0$. So as long the transmitted and received vector agree in strictly more than $k(D + k)$ coordinates, g will be identically zero. Hence, $y - p(x)$ will divide $Q(x, y)$ as long as $t > k(D + k)$. ■

We have now seen that list decoding is possible in polynomial time as long as our errors have a simple algebraic description. We can now remove this final assumption with one last piece of insight.

2.3 List Decoding with Arbitrary Errors

Let's address finally the issue of removing the simplifying assumption from the previous section. Consider an "error set" of values $\alpha_1 \ldots \alpha_e$ and their corresponding received values $r_1 \ldots r_e$, for some $e \leq n$. What is the smallest-degree polynomial $E(x, y)$ which is not identically zero for which $E(\alpha_i, r_i) = 0$ for all $i = 1 \ldots e$?

As an upper bound, degree e would suffice, as we could take $E(x, y) = E_1(x)$, where $E_1(x)$ is a degree e polynomial with zeros at $\alpha_1 \ldots \alpha_e$. Similarly, we could take $E(x, y) = E_2(y)$, where $E_2(y)$ is a degree e polynomial with zeros $r_1 \ldots r_e$. However, we can do much better – we can choose $E(x, y)$ to have degree $\sqrt{e}$ in x and also $\sqrt{e}$ in y. In fact, this result has nothing to do with the fact that the particular set of e values in question represent "errors". We can therefore make the following general claim.

Claim 5 *One can always find a non-zero polynomial $Q(x, y)$ of degree at most $\sqrt{n}$ in both x and y (that is, total degree at most $2\sqrt{n}$), such that $Q(\alpha_i, r_i) = 0$ for all $i = 1 \ldots n$.*

Armed with this extra bit of knowledge, let us now revisit our previous decoding algorithm. When computing Q in step 1, we can now remove the assumption that the degree of Q is "reasonably small", and argue that the degree of Q need not be larger than $2\sqrt{n}$. Therefore, the same analysis tells us that as long as $t > 2k\sqrt{n}$, we can perform list decoding in polynomial time and recover all polynomials p of degree at most k for which $p(\alpha_i) = r_i$ for at least t values of i. This analysis can actually be improved rather easily to $t > 2\sqrt{kn}$. To do so, we construct Q which has degree $\sqrt{kn}$ in x and degree $\sqrt{n/k}$ in y; the same analysis yields the claimed bound. With quite a bit more work, one may improve the bound even further to $t > \sqrt{kn}$, although we won't discuss this now.