6.S897 Algebra and Computation April 30, 2012

Applications of Algebra in Computing: Codes and Decoding

Instructor: Madhu Sudan *Scribe: Michael Forbes*

1 Overview

Today we will cover an application of algebra to computing, specifically talking about the constructions of error correcting codes. Specifically, we aim to cover

- Reed-Solomon Codes
- List decoding of Reed-Solomon codes
- Ideal error correction

We will mention the frontier of this work, encapsulated in folded Reed-Solomon codes, and how this frontier improves on what we show today.

2 Coding Theory

We first motivate coding theory as a topic. Consider the problem of storing data on a CD-ROM (or a modern variant). Such physical devices can be corrupted over time due to mistreatment or decay of physical structure. That is, *errors* will occur in our stored message. Thus, if we want to recover the original message in the future we need to do something to cope with such errors. The main idea is that we will assume that the error rate is somehow bounded, so errors cannot be arbitrarily complicated. In such a situation, we hope to add a small amount of redundancy in our dataset such that this will still allow recovery in the face of errors.

Formally, we have some universe of messages, denoted U, and wish to store it. As is typical, we encode the messages as strings over some finite alphabet Σ. So our encoding map will be $Enc : U \to \Sigma^n$. Typically, we will want Σ to be $\{0, 1\}$. However, other such Σ can be useful if we have a physical medium (such as a CD-ROM) that naturally has a larger Σ. CD-ROM's are this way because their errors come in *bursts* (think of scratches on a CD), so even though the CD is actually encoded in bits, the errors are better modeled as errors on chunks of bits. There are other reasons to take Σ larger. One is that it allows interesting families of codes to be defined, such as the Reed-Solomon codes we shall soon see. The other is that one can often then convert codes over large Σ to codes over $\{0, 1\}$, while retaining interesting properties.

The notion of error we will use is the *Hamming metric*, given by $\Delta(x, y) = |\{i : x_i \neq y_i\}|$, for $x, y \in \Sigma^n$. We shall denote $\mathcal{C} \subseteq \Sigma^n$ as the image of the encoding function Enc, and call $\mathcal{C}$ the code. The *minimum distance* of the code is defined as $\Delta(\mathcal{C}) = \min_{x \neq y \in \mathcal{C}}$. Typically a code will have Σ as some finite field, and oftentimes the universe U can be identified with some Σ^k. In such a situation, we have an $[n, k, d]_q$ code, for a code $\Sigma^k = \mathcal{C} \subseteq \Sigma^n$, with

minimum distance d, over the alphabet $\Sigma = \mathbb{F}_q$. A large minimum distance means that the codewords are well-separated according to the hamming metric, and thus it takes many errors to confuse one codeword with another. Putting such reasoning together, we get

Lemma 2.1 (Hamming)**.** *If the minimum distance of a code $\mathcal{C}$ is d then one can (information theoretically) correct up to $\lfloor (d-1)/2 \rfloor$ errors.*

Note that this decoding is only information theoretic. That is, if there are less than $\lfloor (d-1)/2 \rfloor$ errors then there is a unique codeword that we can decode to. However, the above lemma, which is geometric, does not say how to find that codeword. One of the challenges of coding theory is to develop codes with efficient decoding algorithms. One of the other challenges is to balance the various parameters of interest. That is, we want to maximize the rate k/n of a code for any given minimum distance d. Intuitively, there must be some limit to this rate for any such d, as we must add some redundancy to ensure the distance property. There are many results showing this to be the case. We prove one here, called the Singleton bound (named after an individual named Singleton).

Lemma 2.2 (Singleton Bound)**.** *In an $[n, k, d]_q$ code, we have that $n \geq k + d - 1$.*

Proof. Let $\mathcal{C} \subseteq \Sigma^n$ be an $[n, k, d]_q$ code, so that $|\mathcal{C}| = q^k$. Consider the projection map $\pi : \Sigma^n \to \Sigma^{k-1}$ that takes a codeword and drops the last $n - (k-1)$ symbols. By the pigeonhole principle, we have that there must be $x \neq y \in \mathcal{C}$ such that $\pi(x) = \pi(y)$. Thus x and y agree on the first $k-1$ coordinates, so that $\Delta(x, y) \leq n - (k-1)$. However, $d \leq \Delta(x, y)$, giving the bound. □

Perhaps somewhat surprisingly, this bound is tight, in that Reed-Solomon codes meet it, as we shall now see.

3 Reed-Solomon Codes

Reed-Solomon codes, defined in the 1960's, are one of the most simple applications of algebra to coding theory. Here, we take $\Sigma = \mathbb{F}_q$, for $n \leq q$. Take $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ to be distinct. Our messages $(m_0, \ldots, m_{k-1})$ we be encoded as the evaluations $(p(\alpha_1), \ldots, p(\alpha_n))$, where p is the polynomial $p(x) = \sum_{i=0}^{k-1} m_i x^i$.

To analyze the distance of this code, note that two polynomials of degree $< k$ can agree in at most $k-1$ spots. So two messages m and m' must disagree in at least $n - (k-1)$ evaluations, so that is the distance of the code. We see that this meets the Singleton bound. Codes that meet the Singleton bound are called *maximum distance separable (MDS)* codes, and there are results that show that all such codes are essentially Reed-Solomon codes.

4 List Decoding Reed-Solomon Codes

We now show how to decode Reed-Solomon codes, in a model called *list decoding*. Specifically, we are given points $(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)$ where the α_i are the evaluation points, and the β_i are the corrupted evaluations of our message polynomial. Let k be the degree bound, t the number of errors we allow, and $a = n - t$ be the agreement parameter. We seek to

find polynomials p of degree $< k$ with certain amounts of agreement with the (α_i, β_i) pairs, that is, $|\{i : p(\alpha_i) = \beta_i\}| \geq a$. There are several different regimes where this is interesting.

The first regime is $a = (n + k)/2$. This is the *unique decoding regime*, where there is at most one such polynomial p.

The second regime is $a > \sqrt{2kn}$. In this regime, an inclusion-exclusion type argument (known as the Johnson bound) shows that there are at most $2\sqrt{n/k}$ such polynomials with this level of agreement. The argument relies on the fact that we are asking to fit many polynomials into a smallish hamming ball, but that each polynomial must still be well-separated from each other.

In the regime where $a < \sqrt{kn}$, current techniques can show that there are at most n^2 such polynomials.

We note that the above regimes essentially constitute all of our knowledge about decoding Reed-Solomon codes.

We now comment on why one would want to actually perform list-decoding instead of just unique-decoding. Unfortunately there is no quick black-box motivation, indicating why list-decoding is the correct idea. However, there are black-box applications known. Typically, they use the following idea: list-decoding can still function in the presence of more errors than unique decoding can. Thus, when there are many errors one wants to still be able to do something, and list decoding is that something. Once one gets a list of candidate messages, it is sometimes, but not generically, possible to prune down the candidates to get the correct message. Often, the applications of list-decoding have some auxiliary problem structure that allow this pruning to be done (eg. the messages are all English sentences).

The paradigm of the decoding algorithms is to find a nice algebraic "explanation" for the data. One explanation would be a polynomial $\hat{p}$ such that $\hat{p}(\alpha_i) = \beta_i$ for each i. However, this is not robust to error. For if we start out with a degree $< k$ polynomial p and corrupt some values we might only get a degree n explanation in the above sense. But we really want the explanation to be "low degree", that is, of degree around $k + t$, where t is the number of errors. In what follows, think of $k = n^{1/10}$.

To achieve the robust low-degree explanation, the works of Petersen (1960), Berlekamp/Massey (1972), Welch-Berlekamp (1986) and Gemmel-Sudan (1992) used the idea of a rational function (a ratio of polynomials) instead of just a polynomial. Specifically, they see to get non-zero polynomials A and B such that $B(\alpha_i)\beta_i = A(\alpha_i)$ for all i. The works of Sudan (1997) and Guruswami-Sudan (1998) generalize this to explanations that can depend more than just linearly on the β_i. Specifically, we seek to find a bivariate non-zero polynomial Q such that $Q(\alpha_i, \beta_i) = 0$ for all i.

We can now describe the decoding algorithm (of Sudan):

1. find a non-zero Q such that $Q(\alpha_i, \beta_i) = 0$ for all i, with the total degree of Q to be at most D

2. factor the bivariate polynomial $Q(x, y)$ and output any factor of the form $y - p(x)$, such that p has the desired agreement

Note that the second item is doable, as we saw earlier in the course how to factor bivariate polynomials. As for the first item, note that we are seeking a non-zero solution to a homogeneous system of linear equations, where the variables are the coefficients of Q. By basic

linear algebra, we see that if there are more variables then equations then such a Q will exist, and furthermore is findable by Gaussian elimination. Note that there are n equations and $\binom{D+2}{2}$ variables, so if $D > \sqrt{2n}$ then there is such a non-zero Q. We now need to argue correctness, that is, all such polynomials p with sufficient agreement will appear as a factor $y - p(x)$ of any such Q (note that there could be many such Q and it is not clear which to choose. The following lemma says that any such Q will work). This is given by the following lemma.

Lemma 4.1. *If* $|\{i : p(\alpha_i) = \beta_i\}| \geq a > Dk$ *then* $y - p(x)|Q$, *for any* Q *such that* $Q(\alpha_i, \beta_i) = 0$ *for all* i.

Proof. There are two ways to prove this. The first is to observe that $Q(x, p(x))$ is of degree at most Dk and has more than Dk roots and thus must be identically zero, and as such $y - p(x)$ divides $Q(x, y)$.

Another way is to use Bezout's theorem in the plane. For we have two curves, $y - p(x)$ and $Q(x, y)$ that have more than Dk common zeroes, and Dk is the product of their degrees. This means, by Bezout's theorem, that they share a common factor. However, as $y - p(x)$ is irreducible, the claim follows. □

Note that this allows us to correct from agreement $a \geq k\sqrt{2n}$, and get at most $\sqrt{2n}$ polynomials, as this is the degree of Q and each factor is of degree 1. One can do better by using a notion of "weighted degree". For when we take the polynomial $Q(x, y)$ and substitute $p(x)$ for y we are not using the degrees of x and y in a balanced way. Thus, in constructing Q, one can define the weighted degree of $x^i y^j$ to be $i + kj$ and then ask for Q to be of weighted degree $\leq D$. When doing this, we can then correct from $O(\sqrt{kn})$ agreement, which is stronger than what we derived above.

5 Ideal Error-Correcting Codes

We now abstract the decoding process we just performed, to understand the essence of what was done. We begin by noting that for a polynomial $p(x)$, the residue of p after taking it modulo $x - \alpha$ is just the value $p(\alpha)$. So the Reed-Solomon code can be viewed as taking a polynomial and encoding it by taking it modulo several ideals. We now abstract this.

Let R be a "nice" ring. Let $M \subseteq R$ be a finite subset of R, which will be our message space. Note that R itself might be infinite. Let $I_1, \ldots, I_n$ be ideals of R, such that the quotient rings R/I_j are all finite. That is, the number of distinct equivalence classes $m + I_j$ is finite for each j. Further, we shall assume each such quotient ring is "small", as they will form our alphabet symbols. We shall encode $m \in M$ by the residues $(m + I_1, \ldots, m + I_n)$.

We now apply this to get a new family of codes, called the Chinese Remainder Code. Recall that the rings $\mathbb{F}_q[x]$ and $\mathbb{Z}$ share many properties, and things true in one are usually true in the other. As Reed-Solomon codes operate in the former ring, we seek to find their analogue over $\mathbb{Z}$. The message space over $\mathbb{F}_q[x]$ is the space of all degree $< k$ polynomials, and we can take the message space over $\mathbb{Z}$ to be the numbers $\{1, \ldots, K = 2^k\}$. The ideals over $\mathbb{F}_q[x]$ are the maximal/prime ideals $\langle x - \alpha_i \rangle$ for $\alpha_i \in \mathbb{F}$, so over $\mathbb{Z}$ we can take the maximal/prime ideals $\langle p_i \rangle$ for primes p_i (say the first n primes). Thus, we encode an integer m by $(m \bmod p_1, \ldots, m \bmod p_n)$. Note that if $n \geq k$ then the Chinese Remainder

Theorem tells us that recovery in the presences of no errors is possible, as $\prod_i p_i \geq 2^n \geq 2^k$. Thus, we may naturally ask what happens in the presences of errors, and how we may decode from such errors.

We now talk abstractly about how this decoding may be done, and what properties of the ring we need. The first property we need is a notion of size, such that $size(a+b) \leq size(a) + size(b)$ and $size(ab) \leq size(a)size(b)$. For the integers, the absolute value plays the role of this size function, and for $\mathbb{F}_q[x]$ the function $p \mapsto 2^{\deg(p)}$ will work. We also need that all of the messages in M are "small" with respect to this size measure, and are of size at most K. We will also need that the number of small elements of the ring grows sufficiently fast. The main property that we will use is that if an element a is in all of the ideals $I_1, \ldots, I_n$ then a must either be zero or have large size. As we will restrict ourselves to small messages, this will imply a is zero. This property gives unique decoding of the code with no errors, for two elements a and b cannot be the same modulo all of the ideals, unless $a - b$ is in the intersection of the ideals, but by the above this implies $a = b$.

We now define the decoding problem and show how an analogue of the above list decoding algorithm will solve this new decoding question. We are now given ideals $I_1, \ldots, I_n$ and elements $\beta_1, \ldots, \beta_n$, and we want to find all messages $m \in M$ such that $|\{i : m - \beta_i \in I_i\}| \geq a$. The algorithm is as follows

- Find a non-zero $Q \in R[y]$ that is low-degree in y, and such that all coefficients are of small size, such that $Q(\beta_i) \in I_i$ for all i.
- Factor $Q(y)$ and output all m such that $y - m | Q(y)$ and m satisfies the desired agreement

Again, note that we are factoring polynomials. We saw how to do this in "most" rings, such as over $\mathbb{Z}[y]$ (using the LLL algorithm) in the case of Chinese Remainder codes. We did not cover some rings, such as $\mathbb{F}_q(z)[x]/\langle g \rangle$, for irreducible $g \in \mathbb{F}_q(z)[x]$, but algorithms exist for factoring in such rings.

For the first step, note that for when M is a linear message space, and the notion of small is compatible with this linearity, we can use a linear system solver to find the Q, and this is exactly what was done in the case of Reed-Solomon codes. Over $\mathbb{Z}$, this is more difficult, but still possible. One can observe that the conditions imposed on the polynomial $Q(y)$ are that it is a short-vector in a certain integer lattice. So, just like with factoring in $\mathbb{Z}[y]$, we can use the LLL algorithm to find such polynomials.

Thus, we have seen that the algorithm can be implemented (and returns a small list of polynomials, as $Q(y)$ is of low-degree in y, and each $y - m$ is of degree 1). For correctness, we need to argue that for any appropriate m, $y - m | Q$ for any such Q. To see this, note that $Q(m)$ is small, as m is of small size by assumption and all of the coefficients of Q are small. So then $Q(m)$ is small, and inside the ideals I_i for which $m - \beta_i \in I_i$. As there are at least a such ideals for m, it implies that $Q(m)$ is the intersection of many ideals, so must be either zero or large. But as $Q(m)$ is small, it is then zero. Thus, $y - m | Q(y)$.

We now discuss the parameters of this. Using LLL, we can essentially match the performance of Reed-Solomon codes. More specifically, if we assume that each prime p_i is around $p = n^2$, and take $K = p^k$, and $k = n^{1/10}$, then we can correct from about agreement $\sqrt{2kn}$ or higher.

6 Extensions

We now discuss some extensions to this work. More recent work on Folded Reed-Solomon Codes, due to Guruswami and Rudra (2006), which in turn were based on the work of Paravesh and Vardy (2005), showed how to achieve better rate while doing list-decoding (eg. decoding from $k + \epsilon n$ agreement).

One can also modify the above algorithm on ideal correction to ensure that the $Q(m)$ is in the *product* of many ideals, as opposed to their intersection. This change can make things algebraically nicer, and in turn leads to the study of when a polynomial vanishes with multiplicity. It can be used to get better analysis of list-decoding Reed-Solomon codes.