Algebraic Codes for Error Control

John B. Little

little -at- mathcs -dot- holycross -dot- edu
Department of Mathematics and Computer Science
College of the Holy Cross

SACNAS National Conference
"An Abstract Look at Algebra" – October 16, 2009

Outline

A bit of history

- Beginning of coding theory as a mathematical and engineering subject: a 1948 paper by *Claude Shannon* called "A Mathematical Theory of Communication."
- Shannon lived from 1916 to 2001, and spent most of his working career at Bell Labs and MIT.
- He also made fundamental contributions to cryptography and the design of computer circuitry in earlier work coming from his Ph.D. thesis.

Shannon's conceptual communication set-up

$$\begin{tikzcd} \text{message} \arrow[d] & & \boxed{\text{noise}} \arrow[d] & & \\ \boxed{\text{encoder}} \arrow[r] & \boxed{\text{trans.}} \arrow[r] & \boxed{\text{channel}} \arrow[r] & \boxed{\text{rec.}} \arrow[r] & \boxed{\text{decoder}} \arrow[d] \\ & & & & \text{message} \end{tikzcd}$$

Examples

This is a *very general* framework, incorporating examples such as

- communication with deep space exploration craft (Mariner, Voyager, etc. – the most important early application)
- storing/retrieving information in computer memory
- storing/retrieving audio information (CDs)
- storing/rerieving video information (DVD and Blu-Ray disks)
- wireless communication

Reliability, not (necessarily) secrecy!

- A main goal of coding theory is the design of coding schemes that achieve *error control*: ability to detect and correct errors in received messages.

Reliability, not (necessarily) secrecy!

- A main goal of coding theory is the design of coding schemes that achieve *error control*: ability to detect and correct errors in received messages.
- *Reliability* rather than secrecy

Reliability, not (necessarily) secrecy!

- A main goal of coding theory is the design of coding schemes that achieve *error control*: ability to detect and correct errors in received messages.
- *Reliability* rather than secrecy
- *Cryptography* is the science of designing communications for secrecy, security.

Reliability, not (necessarily) secrecy!

- A main goal of coding theory is the design of coding schemes that achieve *error control*: ability to detect and correct errors in received messages.
- *Reliability* rather than secrecy
- *Cryptography* is the science of designing communications for secrecy, security.
- Definitely related, but not our main focus in this talk!

Error correction

Bienvenqdos a Daljas, SACNAS Cohferenke Atsendeef!

- If you can read this message, then you're doing error correction!

Error correction

Bienvenqdos a Daljas, SACNAS Cohferenke Atsendeef!

- If you can read this message, then you're doing error correction!
- In all human languages, words are usually "far enough apart" that even if some of a message is corrupted, it may still be intelligible

Error correction

Bienvenqdos a Daljas, SACNAS Cohferenke Atsendeef!

- If you can read this message, then you're doing error correction!
- In all human languages, words are usually "far enough apart" that even if some of a message is corrupted, it may still be intelligible
- Usually, only a few legal words that are "close" to what is contained in the received message.

Error correction

Bienvenqdos a Daljas, SACNAS Cohferenke Atsendeef!

- If you can read this message, then you're doing error correction!
- In all human languages, words are usually "far enough apart" that even if some of a message is corrupted, it may still be intelligible
- Usually, only a few legal words that are "close" to what is contained in the received message.
- Robustness in the presence of noise is a very desirable feature that can be "designed in" *using abstract algebra!*

Mathematical setting

- Messages
  - are divided into "words" or blocks of a fixed length, k,
  - use symbols from a finite *alphabet* A with some number q of symbols, typically the *finite field* $\mathbb{F}_q$

Mathematical setting

- Messages
  - are divided into "words" or blocks of a fixed length, k,
  - use symbols from a finite *alphabet* A with some number q of symbols, typically the *finite field* $\mathbb{F}_q$
- Simplest case (also best adapted to electronic hardware) is an alphabet with two symbols: $A = \{0, 1\}$, identified with the finite field $\mathbb{F}_2$ (addition and multiplication *modulo 2* – so 1 + 1 = 0), but we will see others later also.

Mathematical setting

- Messages
  - are divided into “words” or blocks of a fixed length, k,
  - use symbols from a finite *alphabet* A with some number q of symbols, typically the *finite field* $\mathbb{F}_q$
- Simplest case (also best adapted to electronic hardware) is an alphabet with two symbols: $A = \{0, 1\}$, identified with the finite field $\mathbb{F}_2$ (addition and multiplication *modulo 2* – so 1 + 1 = 0), but we will see others later also.
- Usually, all strings or k-tuples in $\mathbb{F}_q^k$ are considered as possible words that can appear in a message.

Encoding and decoding

To correct errors, *redundancy* must included in the encoded message. One way:

- encoded message consists of strings of fixed length $n > k$ over the same alphabet.

Encoding and decoding

To correct errors, *redundancy* must included in the encoded message. One way:

- encoded message consists of strings of fixed length $n > k$ over the same alphabet.
- Then encoding and decoding are functions:

$$E : \mathbb{F}_q^k \to \mathbb{F}_q^n \qquad D : \mathbb{F}_q^n \to \mathbb{F}_q^k$$

where E is 1-1, and $D \circ E = I$ on $\mathbb{F}_q^k$.

Encoding and decoding

To correct errors, *redundancy* must included in the encoded message. One way:

- encoded message consists of strings of fixed length $n > k$ over the same alphabet.
- Then encoding and decoding are functions:

$$E : \mathbb{F}_q^k \to \mathbb{F}_q^n \qquad D : \mathbb{F}_q^n \to \mathbb{F}_q^k$$

  where E is 1-1, and $D \circ E = I$ on $\mathbb{F}_q^k$.
- D might also take a "FAIL" value on some words in the complement of $Im(E)$ containing too many errors to be decodable.

Encoding and decoding

To correct errors, *redundancy* must included in the encoded message. One way:

- encoded message consists of strings of fixed length $n > k$ over the same alphabet.
- Then encoding and decoding are functions:

$$E : \mathbb{F}_q^k \to \mathbb{F}_q^n \qquad D : \mathbb{F}_q^n \to \mathbb{F}_q^k$$

  where E is 1-1, and $D \circ E = I$ on $\mathbb{F}_q^k$.
- D might also take a "FAIL" value on some words in the complement of $Im(E)$ containing too many errors to be decodable.
- $C = Im(E)$ is the *code*. Any such C is a *block code of length n*.

Errors

- Channel errors replace a *codeword* c by a *received word* $x \neq c$.

Errors

- Channel errors replace a *codeword* c by a *received word* $x \neq c$.
- We can think of $x = c + e$, where $e \in \mathbb{F}_q^n$ is the *error vector*.

Errors

- Channel errors replace a *codeword* c by a *received word* $x \neq c$.
- We can think of $x = c + e$, where $e \in \mathbb{F}_q^n$ is the *error vector*.
- The *error weight*

$$\mathrm{wt}(e) = |\{i \mid e_i \neq 0\}|$$

determines how many entries of x are corrupted.

Errors

- Channel errors replace a *codeword* c by a *received word* $x \neq c$.
- We can think of $x = c + e$, where $e \in \mathbb{F}_q^n$ is the *error vector*.
- The *error weight*

$$\mathrm{wt}(e) = |\{i \mid e_i \neq 0\}|$$

  determines how many entries of x are corrupted.
- Example: $\mathrm{wt}(11000101) = 4$.

Errors

- Channel errors replace a *codeword* c by a *received word* $x \neq c$.
- We can think of $x = c + e$, where $e \in \mathbb{F}_q^n$ is the *error vector*.
- The *error weight*

$$\mathrm{wt}(e) = |\{i \mid e_i \neq 0\}|$$

  determines how many entries of x are corrupted.
- Example: $\mathrm{wt}(11000101) = 4$.
- *Decoding* can be seen as finding e (somehow), then subtracting it off to recover c.

Hamming distance

Definition (Hamming Distance)

Let $x, y \in \mathbb{F}_q^n$. *Then*
$d(x, y) = |\{i \in \{1, \ldots, n\} : x_i \neq y_i\}| = \mathrm{wt}(x - y)$.

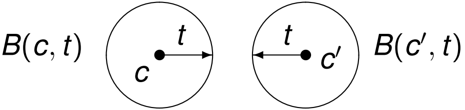
Theorem

Let C *be a code in* $\mathbb{F}_q^n$. *If* $d(c, c') \geq 2t + 1$ *for all distinct* $c, c' \in C$, *then all error vectors of weight* t *or less will be corrected by the "nearest-neighbor" decoding function:*

$$D(x) = E^{-1}(c \in C : d(x, c) \text{ is minimal}).$$

Idea of the proof

Notation: $B(c,t) = \{x \in \mathbb{F}_q^n \mid d(x,c) \leq t\}$ ("Hamming ball")



Condition on $d(c,c')$ implies $B(c,t) \cap B(c',t) = \emptyset$ whenever $c \neq c' \in C$. If c is sent and $\mathrm{wt}(e) \leq t$, then $c+e$ is *still closer to c than it is to any other codeword c'*, and nearest neighbor decoding will correct the error.

Minimum distance

Leads to ...

Definition

Let C be a code in $\mathbb{F}_q^n$. The **minimum distance** *of C, denoted d or $d(C)$, is $d = \min_{c \neq c' \in C} d(c, c')$.*

(That is, d gives the *smallest separation* between any two distinct codewords.) If $d = 13$, for instance, then any error of weight ≤ 6 in a received word can be corrected by nearest-neighbor decoding.

Reed-Solomon – More History

- RS codes are named after Irving Reed and Gustave Solomon.
- Date to 1960, when Reed and Solomon worked at MIT's Lincoln Labs in Massachusetts.
- Reed, who is still living, earned his Ph.D. at Cal Tech and later taught at USC before retiring.
- Solomon, who died in 1996, earned his Ph.D. at MIT, and consulted for many years at JPL in Pasadena.

General Properties

Reed-Solomon codes are codes over an alphabet $\mathbb{F}_q$ (usually $q = 2^r$ for some $r = 4, 8, 16$, etc.) with many good properties:

- They are *linear* – set of codewords is a vector subspace of $\mathbb{F}_q^n$ for $n = q - 1$, and *cyclic* – set of codewords is closed under cyclic shifts

General Properties

Reed-Solomon codes are codes over an alphabet $\mathbb{F}_q$ (usually $q = 2^r$ for some $r = 4, 8, 16$, etc.) with many good properties:

- They are *linear* – set of codewords is a vector subspace of $\mathbb{F}_q^n$ for $n = q - 1$, and *cyclic* – set of codewords is closed under cyclic shifts
- Best possible d for their n and k – meet the *Singleton bound*: $d \leq n - k + 1$

General Properties

Reed-Solomon codes are codes over an alphabet $\mathbb{F}_q$ (usually $q = 2^r$ for some $r = 4, 8, 16$, etc.) with many good properties:

- They are *linear* – set of codewords is a vector subspace of $\mathbb{F}_q^n$ for $n = q - 1$, and *cyclic* – set of codewords is closed under cyclic shifts
- Best possible d for their n and k – meet the *Singleton bound*: $d \leq n - k + 1$
- Good encoding method (via polynomial division)

General Properties

Reed-Solomon codes are codes over an alphabet $\mathbb{F}_q$ (usually $q = 2^r$ for some $r = 4, 8, 16$, etc.) with many good properties:

- They are *linear* – set of codewords is a vector subspace of $\mathbb{F}_q^n$ for $n = q - 1$, and *cyclic* – set of codewords is closed under cyclic shifts
- Best possible d for their n and k – meet the *Singleton bound*: $d \leq n - k + 1$
- Good encoding method (via polynomial division)
- Berlekamp-Massey, Sugiyama (Euclidean Algorithm) decoders – efficiently correct all errors of weight $\leq t$, heavily based on *abstract algebra*

General Properties

Reed-Solomon codes are codes over an alphabet $\mathbb{F}_q$ (usually $q = 2^r$ for some $r = 4, 8, 16$, etc.) with many good properties:

- They are *linear* – set of codewords is a vector subspace of $\mathbb{F}_q^n$ for $n = q - 1$, and *cyclic* – set of codewords is closed under cyclic shifts
- Best possible d for their n and k – meet the *Singleton bound*: $d \leq n - k + 1$
- Good encoding method (via polynomial division)
- Berlekamp-Massey, Sugiyama (Euclidean Algorithm) decoders – efficiently correct all errors of weight $\leq t$, heavily based on *abstract algebra*
- Widely used in applications (e.g. CD audio system, computer memory, etc.)

Constructing RS codes

- Start with the desired *dimension* $k < q$.

Constructing RS codes

- Start with the desired *dimension* $k < q$.
- Let $L_k = \mathrm{Span}\{1, u, u^2, \ldots, u^{k-1}\} \subset \mathbb{F}_q[u]$.

Constructing RS codes

- Start with the desired *dimension* $k < q$.
- Let $L_k = \mathrm{Span}\{1, u, u^2, \ldots, u^{k-1}\} \subset \mathbb{F}_q[u]$.
- We can define a code of dimension k by *evaluating* polynomials $f \in L_k$ to get the codeword entries:

$$\begin{aligned} ev : L_k &\longrightarrow \mathbb{F}_q^{q-1} \\ f &\longmapsto (f(1), f(\alpha), f(\alpha^2), \ldots, f(\alpha^{q-2})) \end{aligned}$$

(where α is a primitive element of $\mathbb{F}_q$, so $\alpha^{q-1} = 1$).

Constructing RS codes

- Start with the desired *dimension* $k < q$.
- Let $L_k = \mathrm{Span}\{1, u, u^2, \ldots, u^{k-1}\} \subset \mathbb{F}_q[u]$.
- We can define a code of dimension k by *evaluating* polynomials $f \in L_k$ to get the codeword entries:

$$\begin{aligned} ev : L_k &\longrightarrow \mathbb{F}_q^{q-1} \\ f &\longmapsto (f(1), f(\alpha), f(\alpha^2), \ldots, f(\alpha^{q-2})) \end{aligned}$$

  (where α is a primitive element of $\mathbb{F}_q$, so $\alpha^{q-1} = 1$).
- The image of *ev* is the RS code – a vector subspace of dimension k in $\mathbb{F}_q^n$ for $n = q - 1$.

$d \Leftrightarrow$ a basic fact for polynomials(!)

- Linearity $\Rightarrow$ in computation of $d(x, y) = \mathrm{wt}(x - y)$, $x - y$ is another codeword.

$d \Leftrightarrow$ a basic fact for polynomials(!)

- Linearity $\Rightarrow$ in computation of $d(x, y) = \mathrm{wt}(x - y)$, $x - y$ is another codeword.
- So, determining $d \Leftrightarrow$ asking how many zeroes can a nonzero polynomial in L_k have?

$d \Leftrightarrow$ a basic fact for polynomials(!)

- Linearity $\Rightarrow$ in computation of $d(x, y) = \mathrm{wt}(x - y)$, $x - y$ is another codeword.
- So, determining $d \Leftrightarrow$ asking how many zeroes can a nonzero polynomial in L_k have?
- The answer is clear – deg $f(u) \leq k - 1$ for all $f(u) \in L_k$, so no more than $k - 1$ roots(!)
- **Proof:** By division, $\beta \in \mathbb{F}_q$ is a root of $f(u)$ $\Leftrightarrow f(u) = (u - \beta)q(x)$. Then $\deg(q(u)) = \deg(f(u)) - 1$. $\square$

$d \Leftrightarrow$ a basic fact for polynomials(!)

- Linearity $\Rightarrow$ in computation of $d(x, y) = \mathrm{wt}(x - y)$, $x - y$ is another codeword.
- So, determining $d \Leftrightarrow$ asking how many zeroes can a nonzero polynomial in L_k have?
- The answer is clear – $\deg f(u) \le k - 1$ for all $f(u) \in L_k$, so no more than $k - 1$ roots(!)
- **Proof:** By division, $\beta \in \mathbb{F}_q$ is a root of $f(u)$ $\Leftrightarrow f(u) = (u - \beta)q(x)$. Then $\deg(q(u)) = \deg(f(u)) - 1$. $\Box$
- Moreover, *some* $f(u)$ of degree $k - 1$ have exactly $k - 1$ distinct roots.

$d \Leftrightarrow$ a basic fact for polynomials(!)

- Linearity $\Rightarrow$ in computation of $d(x, y) = \mathrm{wt}(x - y)$, $x - y$ is another codeword.
- So, determining $d \Leftrightarrow$ asking how many zeroes can a nonzero polynomial in L_k have?
- The answer is clear – $\deg f(u) \le k - 1$ for all $f(u) \in L_k$, so no more than $k - 1$ roots(!)
- **Proof:** By division, $\beta \in \mathbb{F}_q$ is a root of $f(u)$ $\Leftrightarrow f(u) = (u - \beta)q(x)$. Then $\deg(q(u)) = \deg(f(u)) - 1$. $\square$
- Moreover, *some* $f(u)$ of degree $k - 1$ have exactly $k - 1$ distinct roots.
- So minimum weight in $ev(L_{k-1})$ is $d = (q - 1) - (k - 1) = n - k + 1$.

A Recent Development – List Decoding

- Analogy: Say a misprint occurs in something you are reading – “bawn”

A Recent Development – List Decoding

- Analogy: Say a misprint occurs in something you are reading – “bawn”
- Possible corrections changing just one letter: “bawl,” “fawn,” “lawn,” “pawn,” “barn,” etc. – not too many possibilities and maybe can correct from context ...

A Recent Development – List Decoding

- Analogy: Say a misprint occurs in something you are reading – “bawn”
- Possible corrections changing just one letter: “bawl,” “fawn,” “lawn,” “pawn,” “barn,” etc. – not too many possibilities and maybe can correct from context ...
- “Traditional” decoding methods *assume* $d = 2t + 1$ and e with $\mathrm{w}(e) \leq t$ occurs. By Theorem before, $\exists !$ closest codeword c to x and the decoder finds it. (But if an error of weight $> t$ occurs, these often *fail*.)

A Recent Development – List Decoding

- Analogy: Say a misprint occurs in something you are reading – “bawn”
- Possible corrections changing just one letter: “bawl,” “fawn,” “lawn,” “pawn,” “barn,” etc. – not too many possibilities and maybe can correct from context ...
- “Traditional” decoding methods *assume* $d = 2t + 1$ and e with $\mathrm{w}(e) \leq t$ occurs. By Theorem before, $\exists!$ closest codeword c to x and the decoder finds it. (But if an error of weight $> t$ occurs, these often *fail*.)
- List decoding idea is to extend the error weights that can be handled by making decoder output a list of *all* codewords within some *decoding radius* $\tau \geq t$ of x'.

Algebraic Methods – Interpolation

- List decoding algorithms developed by M. Sudan, V. Guruswami, and others.

Algebraic Methods – Interpolation

- List decoding algorithms developed by M. Sudan, V. Guruswami, and others.
- Proceed in two steps after choice of decoding radius τ.

Algebraic Methods – Interpolation

- List decoding algorithms developed by M. Sudan, V. Guruswami, and others.
- Proceed in two steps after choice of decoding radius τ.
- **Interpolation** – First, a two-variable polynomial $Q(u, v)$ is computed to interpolate the received word x: $Q(\alpha^i, x_i) = 0$ for all $0 \leq i \leq q - 1$ (possibly with an associated *multiplicity* m at all of the points).

Factorization

- **Factorization** – Under suitable hypotheses, any polynomial $Q(u, v)$ as in the first step must factor as
$$Q(u, v) = (v - f_1(u)) \cdots (v - f_L(u))R(u)$$
with $\deg f_i(u) \leq k - 1$.
- Once the factorization is found, each $v - f_i(u)$ corresponds to an RS codeword c_i and the algorithm returns the *list* $\{c_1, \ldots, c_L\}$.

Factorization

- **Factorization** – Under suitable hypotheses, any polynomial $Q(u, v)$ as in the first step must factor as
$$Q(u, v) = (v - f_1(u)) \cdots (v - f_L(u))R(u)$$
with $\deg f_i(u) \leq k - 1$.
- Once the factorization is found, each $v - f_i(u)$ corresponds to an RS codeword c_i and the algorithm returns the *list* $\{c_1, \ldots, c_L\}$.
- Again under suitable hypotheses, any RS codeword distance $\leq \tau$ from the received word must appear in the list.

Factorization

- **Factorization** – Under suitable hypotheses, any polynomial $Q(u, v)$ as in the first step must factor as
$$Q(u, v) = (v - f_1(u)) \cdots (v - f_L(u))R(u)$$
with $\deg f_i(u) \leq k - 1$.
- Once the factorization is found, each $v - f_i(u)$ corresponds to an RS codeword c_i and the algorithm returns the *list* $\{c_1, \ldots, c_L\}$.
- Again under suitable hypotheses, any RS codeword distance $\leq \tau$ from the received word must appear in the list.
- Efficient algorithms for both steps are known drawing on techniques from symbolic algebraic computation with polynomials in several variables!

Suggestions For Further Reading

- For More on Coding Theory Basics:
  - W.C. Huffman and V. Pless, *Fundamentals of error-correcting codes*, Cambridge University Press, Cambridge, 2003.
- For More On List Decoding For RS Codes:
  - V. Guruswami, *List Decoding of Error Correcting Codes*, Springer Lecture Notes in Computer Science 3282, Springer-Verlag, Berlin, 2004.
  - T. Moon, *Error Correction Coding*, Wiley-Interscience, Hoboken, 2005.

Thanks for your attention!