Matthew Wright
April 22, 2002
Abstract Algebra – Math 434
Bryan Smith

1 Algebraic Coding Theory

First Half of Day 1
History:

- Algebraic coding theory was created in the late 1940's by Richard Hamming at the Bell Telephone Laboratories
- The first error control codes were single error correcting Hamming codes
- These codes intended for use with long distance telephony
- Convolution codes were next developed in 1955 by P. Elias
- Cyclic codes were developed in 1959
- Algebraic coding theory is now used in compact discs, magnetic tape, modems, fax machines and bar code scanners

Motivation:

- Suppose I want to send one of two signals to a friend across campus
- Together we decide that one shouts will mean that class is cancelled; whereas, two shout will mean that there is class
- I show up to class and the teacher is not there; so I shout twice.
- However, a plane going to McChord AFB is flying by and my friend only hears one shout.
- So he does not go to class :-(
- Thus, we want a means to ensure that the message sent is the same as the received (or at least to know that we received the incorrect message).

General encoding and decoding scheme:
[Message Source]– Message –> [Encoder] –Codeword–>
[Channel] –Received Word–> [Decoder] – Decoded Message –> User
↑
Noise

- From the example we had the problem of noise (a really loud jet).
- Noise is the quintessential problem for coding theory.

Lastly, one must note that algebraic coding theory is different from cryptography.

- Cryptography is the mathematical theory behind sending secret messages
- Algebraic Coding Theory is the mathematical theory of sending messages that arrive with the same content in which they were sent.

Terminology:
Definition: Given finite sets A and B a coding is an onto function $\phi : A \longrightarrow B$
A is called the source alphabet
B is called the code alphabet
If B has two symbols we call B binary
Definition: Given a coding $\phi : A \longrightarrow B$ where $A = \{a_1, ...a_n\}$

then $\phi(a_1), ..., \phi(a_n)$ are called code words and $C = \{\phi(a) : a \in A\}$ is called a code
(Adamek, 5-6)
Definition: V is the space of all n-tuples os 0's and 1's with addition of vectors component wise mod-2 (Pless 2nd Ed, 6)
Definition: C is an $[n, k]$ linear binary code if C is the set of all linear combination of k independent vectors in V. I.e. If C is a k-dimensional subspace of V.(Pless 2nd Ed, 6)
Definition: An $[n, k]$ code over $GF(q)$ is a k-dimensional subspace of F^n, where F^n is the space of all n-tuples with components from $GF(q)$, where $GF(q)$ is the field of order q. (Pless 2nd Ed, 7)
Definition: A Generator Matrix is a matrix whose rows from a basis for an $[n, k]$ code. (Pless 2nd Ed, 7)
Definition: We say that a generator matrix G of an $[n, k]$ code C is in standard from if $G = (I, A)$ where I is the k x k identity matrix and A is a k x $(n - k)$ matrix. (Pless 2nd Ed, 9)
Definition: The weight of a vector u is the number of non-zero components it has and is denoted by $wt(u)$.(Pless 2nd Ed, 10)
Definition: The distance between two vectors u and v, denoted $d(u, v)$, is the number of positions in which they differ.

That is if $u = (u_1, u_2, ..., u_n)$ and $v = (v_1, v_2, ..., v_n)$
Then $d(u, v) = |\{i : u_i \neq v_i\}|$

Note: $d(u, v) = wt(u - v)$
Theorem: The distance function is metric. That is the following properties hold:

For vectors u, v, and $w \in V$
(i) $d(u, v) = 0$ iff $u = v$
(ii) $d(u, v) = d(v, u)$
(iii) $d(u, v) \leq d(u, w) + d(w, v)$ (triangle inequality)

(i) By definition $d(u, v) = 0$ iff $u_i = v_i$ iff $u = v$
(ii) By the definition $d(u, v) = |\{i : u_i \neq v_i\}| = d(v, u)$
The Idea of (iii):
$d(u, v)$ can be thought of as the minimum number of changes of coordinates of u to make it v
So for vectors $u, v, w \in V$
So, to change u to w requires $d(u, w)$ changes of components of u
Likewise, to change w to v requires $d(w, v)$ changes of components of w.
Hence, $d(u, v) \leq d(u, w) + d(w, v)$.
Proof of (iii) :
Let $d(u, w) = a$ and $d(w, v) = c$.
Thus, we have indices $i_1, ..., i_a$ in which u differs from w
And, we have indices $j_1, ..., j_c$ in which w differs from v,
Moreover, $u_i = w_i$ whenever $i \neq i_1, ..., i_a$
And, $w_i = v_i$ whenever $i \neq j_1, ..., j_c$
Consequently, $u_i = v_i$ whenever $i \neq i_1, ..., i_a, j_1, ..., j_c$
Thus, the number of indicies in which u differs from v is at most $a + c$.
Hence, $d(u, v) \leq a + c = d(u, w) + d(w, v)$.
(Theorem and Proof from *Coding Theory*; website)
Day Two:

2 Error Detection

- Error Detection
- Up to this point he have discussed error correction.
- Error Detection is preferred when there is a feedback channel (i.e. the receiver can also communicate with the sender) and retransmission is reasonable.
- The following encoding/decoding schemes are used in situation where we need fast encoding and decoding–since retransmission takes time–and we must be able to apply encoding and decoding methods to words of variable length.

- Theory developed so far relies on the fact that words have a specific length.
- This is not always the case: in Ethernet words range from 480 -12112 bits.
- Also multiplying by a matrix is "harder" computationaly (time and space complexity); hence, we want methods that do not rely on matrix multiplication.

Given $G = \begin{bmatrix} 1\,0\,0\,1\,1\,0 \\ 0\,1\,0\,1\,0\,1 \\ 0\,0\,1\,1\,1\,1 \end{bmatrix}$ the standared generator matrix for a $[6,3]$ linear binary code we encoded the $v = [110]$

By $vG = [110]\begin{bmatrix} 1\,0\,0\,1\,1\,0 \\ 0\,1\,0\,1\,0\,1 \\ 0\,0\,1\,1\,1\,1 \end{bmatrix} = [110011]$

As mentioned the first 3 components consists of the original message and the remaining 3 components consists of redundant information

In general an $[n,k]$ linear code codeword consists of k message components followed by $n-k$ components of redundant information.

2.0.1 Single Even Parity Bit Checking:

This method adds 1 bit of redundant information.
Given $a = (a_1, a_2, ..., a_n)$ a binary vector. (Comer)
Then the codeword of a is $a' = (a_1, a_2, ..., a_n, p)$ where

$p = 1$ if $wt(a)$ is odd or
$p = 0$ if $wt(a)$ is even

Decoding: if the string has an even number of 1's then no error is declared, otherwise an error is declared.
We can also use a generator matrix to accomplish this:
For $a = (a_1, a_2, ..., a_n)$ the generator matrix $G = [I_{nxn,}H]$ where $H = [1\ 1\ ...\ 1]^T$
Example: 1001 encoding simply means appending a 0 since there is an even number of ones.
For $G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$ we can encode 1001 by using the generator matrix G.
If we used the generator matrix encoding scheme this would be "harder" (time complexty and space) then the encoding based on the weight. This example demonstrates the ability to get around having to use matrix multiplication.
Error Analysis

- Observe that if both flipped are 0's or 1's even parity is still preserved.
- Also if a 1 and a 0 is flipped even parity is still preserved.
- This is true of any even number of errors.

2.0.2 Checksums:

(Following from *Comer*)
Basic idea: The sender treats the message as binary integers and computes the sum in specific increments and appends the total to the end of the message.
Example: Assume that the interval is 4 bits. The check sum for 000100111001 is computed as follows:
Splitting the message into 4 bit increments and computing the value for each set we have:
0001 = 1 (this is base 10)
0011 = 3
1001 = 9
The checksum is 13, and the binary representation of 13 is 1101. So the codeword is (the bold represents the checksum): 0001001110011**1101**

- Observe that 100100110001 also has the check sum of 1101.
- I will not prove this, but it can be show that check sums do not detect most common kinda of errors, so we need a better method.

- We will now look at Cyclic Redundancy Check as an example of such a method.

2.0.3 Cyclic Redundancy Check (CRC):

We must first observe that we can think of words as polynomials.
For the vector $v = (a_n, a_{n-1}, a_{n-2}, ..., a_0)$ with components from Z_2. We can think of v as corresponding to the polynomial $f(x) = a_n x^n + a_{n-1}x^{n-1} ... + a_1 x + a_0$ in $Z_2[x]$
Example: $(1, 0, 1, 1, 0, 1)$ corresponds to $g(x) = x^5 + x^3 + x^2 + 1$
Definition: $f(x) = a_n x^n + a_{n-1}x^{n-1} ... + a_1 x + a_0 \in Z[x]$ is primitive if its coefficients have no common factors other then +1 or -1 and the leading coefficient is positive. (*Class Notes*)

Encoding for Cyclic Redundancy Check: Let $a_n a_{n-1} ... a_0$ be a binary message i.e. $a_i \in Z_2$ and $m(x)$ correspond to the vector $(a_n, a_{n-1}, a_{n-2}, ..., a_0)$
i.e. $m(x) = a_n x^n + a_{n-1}x^{n-1} ... + a_1 x + a_0$
Given a primitive polynomial $g(x) \in Z_2[x]$–called the generator polynomial–such that $\deg(g(x)) = k$ we compute the CRC as follows:
Multiply $m(x)$ by x^k
Then, by the division algorithm $m(x)x^k = g(x)q(x) + r(x)$ such that $\deg(r) = 0$ or $\deg(r) \leq k$
The polynomial for the transmitted code word is $t(x) = m(x)x^k + r(x)$
So, $r(x) = r_k x^k + ... + r_0$
The code word itself has the form: $(a_n, a_{n-1}, a_{n-2}, ..., a_0, r_k, r_{k-1}, ..., r_0)$

Detecting Errors with Cyclic Redundancy Check: *Note: for $r(x) \in Z_2[x]$ then $r(x) + r(x) = 0$*
Example: $(x^3 + x + 1) + (x^3 + x + 1) = 2x^3 + 2x + 2 = 0x^3 + 0x + 0 = 0$
If $g(x)$ divides $t(x)$ without remainder then no errors have occurred.
Proof:

We assume that $t(x)$ was transmitted without error.
$t(x) = m(x)x^k + r(x)$ by the definition of $t(x)$
$$\begin{aligned} &= m(x)x^k + r(x) \\ &= g(x)q(x) + r(x) + r(x) \\ &= g(x)q(x) + 2(r(x)) \\ &= g(x)q(x) + 0 \end{aligned}$$
Thus, $g(x)$ evenly divides $t(x)$

Example: (*The Cyclic Redundancy Check*; website)
Suppose we want to send 11010111 where the generator polynomial $g(x) = x^3 + x^2 + 1$
Then $m(x) = x^7 + x^6 + x^4 + x^2 + x + 1$
So $m(x)x^3 = x^{10} + x^9 + x^7 + x^5 + x^4 + x^3$
Using polynomial long dividing $m(x)x^3$ by $g(x)$.

$$\begin{array}{r} x^7 + \quad + x^2 + 1 \\ \hline x^3 + x^2 + 1 \;)\; x^{10} + x^9 + x^7 + x^5 + x^4 + x^3 \\ \quad x^{10} + x^9 + x^7 \\ \hline \quad x^5 + x^4 + x^3 \\ \quad x^5 + x^4 + 0x^3 + x^2 \\ \hline \quad x^3 + x^2 \\ \quad x^3 + x^2 + 1 \\ \hline \quad 1 \end{array}$$

Thus, $r(x) = 1$
So, $t(x) = x^{10} + x^9 + x^7 + x^5 + x^4 + x^3 + 1$
Hence, the code word for 11010111 is 11010111001
Looking at the binary string itself we have:

```
      _____1 0 0 0 0 1 0 1
1101) 1 1 0 1 0 1 1 1 0 0 0
        1 1 0 1 0 0 0 0 0 0 0
        ---------------------
        0 0 0 0 0 1 1 1 0 0 0
          0 0 0 0 0 1 1 0 1 0 0
          -------------------
          0 0 0 0 0 0 0 1 1 0 0
            0 0 0 0 0 0 0 1 1 0 1
            ---------------------
                                1
```

Show addition in Z_2 and logical XOR

- Shift the binary representation of the generator polynomial so that it has the same number of bits as the binary message
- XOR the two
- Shift the generator polynomial over until its degree is the same as the polynomial after the XOR, and continue with step 2.
- Do this until the remainder is left.

Mention difficulty in matrix multiplication.
So this method is much easier then using a generator matrix.

How to Chose the Generator Polynomial: **Claim**: For a generator polynomial of $g(x)$ where $\deg(g(x)) = k$, we can choose $g(x)$ such that it is possible to detect:

(i) all single errors.
(ii) all double errors.
(iii) any odd number of errors.
(iv) burst-errors of length less then the degree of $g(x)$.

Let $e(x)$ correspond the error that has occurred during transmission.
So the codeword received is $t'(x) = t(x) + e(x)$ where $t(x)$ is the transmitted code.
We want to chose $g(x)$ such that it is not a factor of $t'(x)$
Since $g(x)$ divides $t(x)$ we want to chose $g(x)$ so that it does not divide $e(x)$
(The following proofs were adapted from: *Cyclic Codes and the CRC (Cyclic Redundancy Check) Codes, and Overview of Error Detection and Correction*)

Single Errors: Let $g(x) = x^k + ... + 1$ and let $e(x)$ represent a single error.
So $e(x) = x^i$ for $i \in \{0, 1, 2, 3, ...\}$
Observe that $g(x)h(x)$ will have at least two terms $\forall\ h(x) \neq 0 \in Z_2[x]$
Hence, $g(x)h(x) \neq x^i\ \ \forall\ h(x) \in Z_2[x]$ and $i \in \{0, 1, 2, 3, ...\}$
$\Rightarrow g(x)h(x) \neq e(x)\ \ \forall\ h(x) \in Z_2[x]$
Thus, $g(x)$ does not divide $e(x)$.
So choosing $g(x)$ to have at least the terms x^k and 1 ensures that we can detect all single errors.

All Odd Number of Errors: Let $g(x)$ have $(x + 1)$ as a factor; then we can detect all odd number of errors
For $g(x) = (x + 1)q(x)$ and let $e(x)$ representing an odd error pattern.
Suppose $g(x)$ divides $e(x)$.
Then $e(x) = (x + 1)q(x)p(x)$
Observe that $e(1) = 1$ since e has an odd number of terms
However $e(1) = (1 + 1)q(1)p(1) = 0(q(1)p(1)) = 0$
So $0 = e(1) = 1$
A contradition, hence $g(x) = (x + 1)q(x)$ does not divide $e(x)$
Thus choosing $g(x)$ such that it has a factor of $(x + 1)$ ensures that all odd number of errors can be detected.

Double Errors: (On Handout) So $e(x) = x^i + x^j$ where $(\deg(e(x)) - 1) \geq i > j \geq 0$
Factoring gives us $e(x) = x^j(x^{i-j} + 1)$
Thus we want to choose $g(x)$ such that it does not divide x^j or $x^m + 1$ where $0 \leq m \leq \deg(e(x))$
If we have chosen $g(x)$ to detect all single errors then $g(x)$ will not divide x^j.
So how do we choose $g(x)$ such that it does not divide $x^m - 1$?
Claim: For $p(x)$ a primitive polynomial of degree N, the smallest a for which $p(x)$ will divide $x^a + 1$ is $a = 2^N - 1$
Thus, we choose $g(x)$ such that its degree, k, satisfies $m = 2^k - 1$, then $g(x)$ will not divide $x^m + 1$
Hence, chosing $g(x)$ such that it contains at least the factors x^k and 1, and its degree, k, satisfies $m = 2^k - 1$ where m is the degree of the largest possible $e(x)$ ensures that all double errors can be detected.

Any Burst Error of length $\leq \deg(g(x)) = k$ Let $e(x)$ correspond to a burst error of length $p \leq k$
Then $e(x) = x^i(e'(x))$ where $e'(x) = x^{p-1} + x^{p-2} + ... + 1$
So $g(x)$ does not divide $e'(x)$ since $\deg(e'(x)) < \deg(g(x))$
Also, $g(x)$ does not divide x^i if it detects single errors.
Thus, any $g(x)$ that detects single errors will also detect any burst error of length $\leq k$.
The following is a list of generator polynomials used in popular protocols:
CCITT Polynomial: $x^{16} + x^{12} + x^5 + 1$
CRC-16 Polynomial: $x^{16} + x^{15} + x^2 + 1$
CRC-12 Polynomial: $x^{12} + x^{11} + x^3 + x + 1$
AUTODIN-II Polynomial (32-bit): $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (*Flannery*)

Any questions?

References: Gallian, Joseph A. *Contemporary Abstract Algebra. 4th ed.* (Houghton Mifflin, 1998)
Pless, Vera. *An Introduction to the Theory of Error Correcting Codes Theory. 2nd ed.* (John Wiley & Sons Inc., 1989)
Flannery, Brian P. et al. *Numerical Recipes in C: The Art of Scientific Computing Second Edition* (Cambridge University Press, 1988)
Comer, Dougla E. *Computer Networks and Internets with Internet Applications.* 3rd ed. (Prentice Hall, 2001)
Coding Theory
<http://www.maths.sussex.ac.uk/Staff/JWPH/TEACH/CODING02/notes.pdf>
The Cyclic Redundancy Check
<http://www.cs.williams.edu/~tom/courses/336/outlines/lect7_2.html>
Cyclic Codes and the CRC (Cyclic Redundancy Check) Codes
<http://www.seas.upenn.edu/~kassam/tcom370/n99_9.pdf>
Overview of Error Detection and Correction
<http://netlab.ece.iupui.edu/classes/2001-spring/EE547/03-Digital%20Transmission%20Fundamentals%202(3).pdf>