

## ECE 7670

### Lecture 9 – Introduction to multiple-error correcting codes

**Objective:** To demonstrate important concepts that lead to more general families of codes; to introduce the idea behind Galois fields.

We have seen how the Hamming code can be used to correct single errors, and met syndrome decoding and standard arrays. We have also seen that the storage and/or decoding complexity could be very high. What is needed to deal with this complexity is some sort of structure. While linear algebra provides some structure, more algebraic structure is needed.

The material in these lecture notes is drawn from Elwyn Berlekamp, *Algebraic Coding Theory* (McGraw-Hill, 1968). The book is definitely a classic in coding, and should be explored by all who are truly interested in this material. Of this material, Berlekamp writes: “The student is urged to study the next section with care. In my opinion, it is the most important section of the book.”

First, a reminder: For a Hamming code, we can look at the syndrome and determine, from the column corresponding, what the error location is. For example, if we write

$$H = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \cdots \quad \mathbf{h}_n]$$

the syndrome  $\mathbf{r}H^T = \mathbf{h}_i$ , where  $i$  indicates the location of the error (assuming, of course, that there is only one).

Let us take the case of a (31, 26) Hamming code, and write the parity check matrix as

$$H = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 0 & 1 & \cdots & 0 & 1 \end{bmatrix}$$

That is, for this example we order the columns in binary numerical order. We will consider each column as a 5-tuple, and also write this as

$$H = [\gamma_1 \quad \gamma_2 \quad \gamma_3 \quad \cdots \quad \gamma_{30} \quad \gamma_{31}]$$

where the  $i$ th column of  $H$  has the binary representation of  $i$  in the symbol  $\gamma_i$ .

Now, let us move beyond Hamming codes and attempt to formulate a double error correcting code. We will do this (in this example) by starting with the same parity check matrix, but not adding 5 more rows. We will denote this as

$$H = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 0 & 1 & \cdots & 0 & 1 \\ f_1(1) & f_1(2) & f_1(3) & \cdots & f_1(30) & f_1(31) \\ f_2(1) & f_2(2) & f_2(3) & \cdots & f_2(30) & f_2(31) \\ f_3(1) & f_3(2) & f_3(3) & \cdots & f_3(30) & f_3(31) \\ f_4(1) & f_4(2) & f_4(3) & \cdots & f_4(30) & f_4(31) \\ f_5(1) & f_5(2) & f_5(3) & \cdots & f_5(30) & f_5(31) \end{bmatrix}.$$

The argument tells which column we are dealing with, and the subscript tells which “bit”. Another way to think about this that is notationally important is to have a function

$$f = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix}$$

which maps binary 5-tuples to binary 5-tuples.

In the shorthand notation, we could write

$$H = \begin{bmatrix} \gamma_1 & \gamma_2 & \gamma_3 & \cdots & \gamma_{30} & \gamma_{31} \\ f(\gamma_1) & f(\gamma_2) & f(\gamma_3) & \cdots & f(\gamma_{30}) & f(\gamma_{31}) \end{bmatrix}$$

We might have, for example,  $f(\gamma) = \gamma$ , or  $f(\gamma) = (1, 1, 1, 1, 1) + \gamma$ . To do much more than this simple stuff, however, we need some way of dealing with these 5-tuples as objects in their own right. We need to develop and appropriate arithmetic: add, subtract, multiply, divide. Adding is easy — we simply go component by component. But how do we multiply?

The key here is to think of each 5-tuple as corresponding to a polynomial of degree  $\leq 4$ . For example:

$$\begin{aligned} (0, 0, 0, 0, 0) &\leftrightarrow 0 \\ (0, 0, 0, 0, 1) &\leftrightarrow 1 \\ (0, 0, 0, 1, 0) &\leftrightarrow x \\ (0, 0, 1, 0, 0) &\leftrightarrow x^2 \\ (1, 0, 1, 0, 1) &\leftrightarrow x^4 + x^2 + 1 \end{aligned}$$

Note that each coefficient of the polynomials is binary; we assume that addition is modulo 2 (i.e., over GF(2)). Clearly, addition of polynomials accomplishes exactly the same thing as addition of the vectors. (They are **isomorphic**.)

How can we multiply? We want our polynomials representing the 5-tuples to have degree  $\leq 4$ , and yet when we multiply, the degree may exceed that. For example,

$$(x^3 + x + 1)(x^4 + x^3 + x + 1) = x^7 + x^6 + x^5 + 3x^4 + 2x^3 + x^2 + 2x + 1 = x^7 + x^6 + x^5 + x^4 + x^2 + 1.$$

To reduce the degree, we choose some polynomial  $M(x)$  of degree 5, and **reduce the product modulo**  $M(x)$ . That is, we divide by  $M(x)$  and take the remainder. (Analogous to multiplication modulo and integer.)

Let us take  $M(x) = x^5 + x^2 + 1$ . Performing the division, we get a quotient of  $x^2 + x + 1$ , and remainder of  $x^3 + x^2 + x$ . Then we can write

$$\begin{aligned} (x^3 + x + 1)(x^4 + x^3 + x + 1) &= x^7 + x^6 + x^5 + x^4 + x^2 + 1 \\ &\equiv x^3 + x^2 + x \pmod{x^5 + x^2 + 1}. \end{aligned}$$

In general, we write

$$A(x) \equiv a(z) \pmod{M(x)}$$

iff there is a polynomial  $q(x)$  such that

$$A(x) = q(x)M(x) + a(x).$$

Now, before we proceed, some facts about modulo operations (these go way back to Gauss)

- If

$$a(x) \equiv A(x) \pmod{M(x)}$$

and

$$b(x) \equiv B(x) \pmod{M(x)}$$

Then

$$a(x) \pm b(x) \equiv A(x) \pm B(x) \pmod{M(x)}$$

$$a(x)b(x) \equiv A(x)B(x) \pmod{M(x)}.$$

Our modulo operations allows us now to add, subtract, and multiply these 5-tuples, considered as polynomials modulo some  $M(x)$ . Can we divide? More fundamentally, given a polynomial  $a(x)$ , is there some other polynomial  $A(x)$  — we may consider it a multiplicative inverse or a reciprocal — such that

$$a(x)A(x) \equiv 1 \pmod{M(x)}.$$

The answer lies in the oldest algorithm in the world. (More details later!)

For polynomials  $a(x)$  and  $b(x)$ , there exist polynomials  $A(x)$  and  $B(x)$  such that

$$a(x)A(x) + b(x)B(x) = (a(x), b(x))$$

where  $(a(x), b(x))$  is the standard notation for the greatest common divisor of  $a(x)$  and  $b(x)$ . Does this provide the answer. Well, suppose that  $(a(x), b(x)) = 1$ ; then we have

$$a(x)A(x) + b(x)B(x) = 1$$

We can then say that

$$a(x)A(x) \equiv 1 \pmod{b(x)}$$

So if we put  $M(x)$  in the role of  $b(x)$ , we have the answer. In order to have this work for any  $a(x)$ , we need to ensure that  $(a(x), M(x)) = 1$  for any  $a(x)$ . This means that we need  $M(x)$  to *have no factors* — it is irreducible. (This is quite analogous to being a prime number.) [Show that  $M(x) = x^5 + x^2 + 1$  is irreducible.]

Observe that when we have  $a(x)A(x) \equiv 1 \pmod{M(x)}$ , we can say that division by  $a(x)$  is equivalent to multiplication by  $A(x)$  (when operations are done modulo  $M(x)$ ).

Now, let us suppose that there are two errors, occurring at positions  $i_1$  and  $i_2$ . Since the code is linear we can suppose that we can write

$$\mathbf{r} = (0, 0, \dots, \underbrace{1}_{i_1}, \dots, \underbrace{1}_{i_2}, 0, \dots, 0)$$

We find

$$\mathbf{r}H^T = (s_1, s_2)$$

with

$$\gamma_{i_1} + \gamma_{i_2} = s_1$$

$$f(\gamma_{i_1}) + f(\gamma_{i_2}) = s_2$$

We will write this as

$$\gamma_1 + \gamma_2 = s_1$$

$$f(\gamma_1) + f(\gamma_2) = s_2$$

If we set our function up correctly, we will have two equations in two unknowns, which we could solve for  $\gamma_i$  which, in turn, will determine  $i$  (since  $\gamma_i$  is, in this case, simply the binary representation of  $i$ ).

Let us consider some possible simple functions. One might be a simple multiplication:  $f(\gamma(x)) = a(x)\gamma(x) \pmod{M(x)}$ . But this would lead to a dependent equation:  $s_2 = as_1$ ; the new parity check equations would tell us nothing new.

We could try  $f(\gamma) = \gamma + a$ ; This would not help, since we would always have  $s_2 = s_1$ .

Let us try some powers. Say  $f(\gamma) = \gamma^2$ . We would then obtain

$$\gamma_1 + \gamma_2 = s_1$$

$$\gamma_1^2 + \gamma_2^2 = s_2.$$

This looks like independent equations, but we have to remember that we are dealing with operations modulo 2. Notice that

$$s_1^2 = (\gamma_1 + \gamma_2)^2 = \gamma_1^2 + \gamma_2^2 + 2\gamma_1\gamma_2 = \gamma_1^2 + \gamma_2^2 = s_2.$$

We have only the redundant  $s_1^2 = s_2$ : the second equation is the square of the first, and conveys no new information.

Try  $f(\gamma) = \gamma^3$ . Now the decoder equations are

$$\gamma_1 + \gamma_2 = s_1$$

$$\gamma_1^3 + \gamma_2^3 = s_2$$

These are independent! Now let's see what we can do to solve this. We can do more or less conventional algebraic manipulation (keeping in the back of our mind how we do multiplication and division).

We can write

$$s_2 = \gamma_1^3 + \gamma_2^3 = (\gamma_1 + \gamma_2)(\gamma_1^2 - \gamma_1\gamma_2 + \gamma_2^2) = s_1(\gamma_1^2 + \gamma_1\gamma_2 + \gamma_2^2) = s_1(\gamma_1\gamma_2 - s_1^2)$$

Hence we have the two equations

$$\gamma_1 + \gamma_2 = s_1$$

$$\gamma_1\gamma_2 = s_1^2 + \frac{s_2}{s_1}$$

if  $s_1 \neq 0$ . (Note that the first deals with sums of values, and the second with products.) We can combine these into a quadratic:

$$\gamma(s_1 + \gamma) = s_1^2 + \frac{s_2}{s_1}$$

or

$$\gamma^2 + s_1\gamma + \left(s_1^2 + \frac{s_2}{s_1}\right) = 0$$

or

$$q(x) = 1 + s_1\gamma^{-1} + \left(s_1^2 + \frac{s_2}{s_1}\right)\gamma^{-2} = 0.$$

For reasons hopefully to be made clear later, it is more useful to deal with the reciprocals of the roots. The polynomial  $q(x)$  is said to be an error locator polynomial. The roots of the quadratic  $q(x)$  can be reciprocated to find  $\gamma_1$  and  $\gamma_2$ . (How to find the roots?)

If there is only one error, then  $\gamma_1 = s_1$  and  $\gamma_1^3 = s_2$ , and we end up with the equation  $1 + s_1\gamma^{-1} = 0$ . And if there are no errors, then  $s_1 = s_2 = 0$ .

Let us summarize the steps we have taken. First, we have devised a way of operating on 5-tuples as single algebraic objects, defining addition, subtraction, multiplication, and division. This required finding some irreducible polynomial  $M(x)$  which works behind the scenes. Once we have got this, the steps are as follows:

1. We compute the syndrome  $\mathbf{r}H^T$ .
2. From the syndrome, we set up the error locator polynomial. If we let

$$S_1 = \sum \gamma_i = s_1 \quad S_2 = \sum \gamma_i^3 = s_2$$

we can write

$$q(z) = 1 + S_1z + \left(S_1^2 + \frac{S_2}{S_1}\right)z^2$$

We note that there must be some relationship between the sums of the powers of roots and the coefficients.

3. We then find the roots of the polynomial, which determine the error locations.

For binary codes, knowing where the error is is sufficient to knowing how to correct it. For nonbinary codes (which are *very* commonly used), there is another step: knowing the error location, we must also determine the error value. This involves setting up another polynomial, the error evaluation polynomial, whose roots determine the error values.

The above steps establish the outline for the next several weeks. Not only will we develop more fully the arithmetic, but we will be able to generalize to whole families of codes, capable of correcting many errors. However, the concepts are all quite similar to those demonstrated here.

It is historically interesting that it took roughly ten years of research to bridge the gap between Hamming and the code presented above. Once this was accomplished, other generalizations followed quickly.