Polynomial Codes

Shan-Yuan Ho

April 24, 2010

In the last section, we saw that for arbitrary linear codes, we required an encoder/generator matrix G to specify the code. For single error correction, the parity check matrix H was determined from G and used in the decoding process. A special class of linear codes are polynomial codes. For correcting single errors, one big advantage of polynomial codes is that specifying only the first row of the encoder/ generator matrix G is sufficient to describe the entire code.

Algebraic coding theory is based on the structure of finite fields, which we have studied previously. We need to know a bit more of algebra for the error correcting codes we are about to study, so we first discuss them.

1 Polynomial Rings

Denote the finite field $(p, +, \cdot)$ as $GF(p)$. A *polynomial* over a field $GF(p)$ is of the form

$$f(x) = c_0 + c_1 x + c_2 x^2 + \cdots c_{n-2} x^{n-2} + c_{n-1} x^{n-1} \tag{1}$$

where the coefficients $c_0, c_1, c_2, \ldots, c_{n-1}$ are elements of $GF(p)$.

A *monic* polynomial is a polynomial where the highest order term c_{n-1} equal to 1. A *monomial* is a polynomial with only one term and has the form x^n for some integer $n \geq 0$.

Polynomial Rings are analagous to the ring of integers. A polynomial $p(x)$ is *divisible* by a polynomial $q(x)$ if there exists a polynomial $r(x)$ such that $p(x) = q(x)r(x)$. The polynomials $q(x)$ and $r(x)$ are also called *factors* of $p(x)$. A polynomial $p(x)$ that is only divisible by α or $\alpha p(x)$ for some $\alpha \in GF(p)$ is called an *irreducible polynomial.* A *prime polynomial* is a monic irreducible polynomial of degree at least 1. In other words, a prime polynomial cannot be factored into two polynomials of smaller degree.

Primality depends on the field that the coefficients are defined over. For example, $1 + x^2$ is prime if the coefficients are defined over the rationals. However, if the coefficents are from the binary field $GF(2)$, then $1 + x^2$ is not prime because it factors into $(1 + x)(1 + x)$.

Recall that a *primitve element* $\alpha \in GF(q)$ is such that every element in the field can be expressed as a power of α, so primitive elements are useful for constructing fields. Similarily, a *primitive polynomial* $p(x)$ of degree r with coefficients in a field F has the property that the remainders of monomials upon dividing by $p(x)$ include every non-zero polynomial of degree up to $r - 1$. For every degree, there is at least one (in fact, there are generally many) primitive polynomials of that degree over $GF(q)$.

2 Single Error Correcting Polynomial Codes

For linear codes, we represented our messages, codewords, and received messages as vectors. We will now represent these vectors as coefficients of polynomials defined over the binary field $GF(2)$ (that is, $0+0 = 1+1 = 0$ and $0+1 = 1+0 = 1$). An example of polynomial addition and multiplication is as follows. The addition of binary sequences 101 + 110 is represented as $(1+x^2)+(1+x) = x+x^2$ which is 011. The multiplication of sequences 1011×11 gives $(1+x^2+x^3)\times(1+x) = 1+x+x^2+x^4$ which is 11101.

Our message $m(x)$ is assumed to be have k binary bits, which we will represent as a polynomial of degree $k - 1$. The j-th bit is the coefficient of x^{j-1} for each j, $1 \leq j \leq k$. The codeword $c(x)$ for each message $m(x)$ is generated by multiplying $m(x)$ by a fixed generator polynomial $P(x)$. If $P(x)$ has degree r, then the codeword has length $n = k + r$.

$$c(x) = m(x)P(x) = \sum_{i=1}^{n} c_i x^i \tag{2}$$

This encoding operation is a linear code since $[m(x) + m'(x)]P(x) = m(x)P(x) + m'(x)P(x)$.

The basis for this code is the set of monomials $1, x, x^2, x^3, \ldots, x^k$, which corresponds to the binary sequences $(100\cdots0), (0100\cdots0), (0010\cdots0), \ldots, (00\cdots001)$. These are the weight 1 message words. To encode a monomial x^j, we multiply it by $P(x)$ and notice that it merely increases all the exponents of $P(x)$ by j.

The matrix form of a polynomial code is that each row is a cyclic shift (one step to the right) of the previous row, since the lower row is x times the previous row. Thus, to specify the generator matrix of this linear code, all we need to know is the first row, which is $P(x)$. We can see such an example below for a $(7, 4)$ code.

Since our code can correct at most one error, we will assume our received sequence has either 0 errors or 1 of the n bits in error. If there are 2 or more errors, then we are doomed. We will represent the error by the *monomial* $e(x)$ where

$$e(x) = x^j \qquad 1 \leq j \leq n \tag{3}$$

if the bit error occurred in the j-th bit position of the received sequence $r(x)$. If no errors occurred, then $e(x) = 0$. Thus,

$$r(x) = c(x) + e(x) = m(x)P(x) + e(x) \tag{4}$$

We then divide $r(x)$ by $P(x)$ to retrieve message $m(x)$. Suppose there is one error in the j-th position. Then upon dividing $r(x)$ by $P(x)$, we get

$$\text{Rem}[\,r(x)\,] = \text{Rem}[\,e(x)\,] = \text{Rem}[\,x^j\,] \qquad (\text{mod } P(x)) \tag{5}$$

because $c(x)$ is divisible by $P(x)$.

We will be able to determine j, the bit position in error, if and only if each monomial x^j in the code has a unique remainder upon dividing by $P(x)$. Recall that with binary field F, there are $2^r - 1$ non-zero polynomials of maximum degree $r - 1$. If we use a primitive polynomial as the generator polynomial of a code of length $n = 2^r - 1$, then every non-zero remainder will map to the remainder of some monomial. Note that this means any codeword of length $2^r - 1$ produces the same remainder as some monomial. Therefore, it is within distance 1 of a codeword. This implies that our polynomial code is a *perfect* single error correcting code!

Finding a perfect single error correcting polynomial code with r check bits is equivalent to finding a primitive polynomial of degree r.

3 The Generator Polynomial $P(x)$

We now investigate how to find the generator polynomials of perfect single error correcting codes. This means that the polynomials must be primitive polynomials.

By convention, we mapped a binary sequence to a polynomial from left to right. We could also have mapped the sequence in reverse from right to left. The reverse of the binary sequence 100101 is 101001. The corresponding polynomial of our sequence is $1+x^3+x^5$ and its *reverse polynomial* is $1+x^2+x^5$. The properties of primality and primitivity are preserved for reverse polynomials. In other words, if a polynomial is prime, its reverse is also prime. The reverse polynomial of a primitive polynomial is also primitive. Homework: prove this.

Table 1: Remainder Table for $1 + x + x^2$

Monomial	Remainder (mod $p(x) = 1 + x + x^2$)
1	1
x	x
x^2	$1 + x$
x^3	1

- *Primitive polynomials of degree 1.* There are exactly two of them, namely, x and $x+1$. Multiplying any message by x merely translates the entire message by one location and places a zero at the begininng, which is not very useful in error correction. Note that $1 + x$ is the factor of any polynomial with an even number of terms, since $P(1) = 0$. Thus, 1 is a root of the polynomial and $1 - x$ (same as $1 + x$) is a factor of the polynomial.

  Thus, multiplying any message by $(1 + x)$ separates the codewords to a minimum distance of 2. This means that it can detect one error but cannot correct it. The code generated from this generator polynomial $P(x)$ is called a *parity check code.*

- *Primitive polynomials of degree 2.* There must be a constant term and a quadratic term, otherwise it would be divisible by x. It must also have an odd number of terms, otherwise it would be divisible by $1 + x$. Thus, there is only one primitive polynomial of degree 2, namely, $1 + x + x^2$. Its remainder table is

- *Primitive polynomials of degree 3.* There are exactly 2 of them, $1+x+x^3$ and $1 + x^2 + x^3$ and the reverses of each other. They are generator polynomials of a $n = 7$ bit single error correcting code for message length $k = 4$.

- *Primitive polynomials of degree 4.* There must be a constant term and an odd number of terms. $1 + x^2 + x^4 = (1 + x + x^2)^2$ and $(1 + x + x^2 + x^3 + x^4)$ are not primitive. There are two polynomials left, $(1 + x + x^4)$ and $(1 + x^3 + x^4)$, which are primitive and generate perfect codes of length $n = 2^4 - 1 = 15$ bits.

By similar procedures, we can find prime and primitive polynomials of higher degrees; (i) write down all the polynomials of the degree that have a constant term and an odd number of terms; (ii) remove all polynomials divisible by lower degree polynomials; (iii) generate the remainder tables of the remaining polynomials and see if it is primitive. Primitive polynomials exist for every degree over every finite field.

To test whether a polynomial is primitive, we can compute the *remainder table.* This is a table giving, for each monomial from $x^0 = 1$ and x^{2^d-2}. One way to do this would be to use long division to find the remainder when each of these polynomials

Table 2: Remainder Table for $1 + x^2 + x^3$

Monomial	$(\text{mod } p(x) = 1 + x^2 + x^3)$
1	1
x	x
x^2	x^2
x^3	$1 + x^2$
x^4	$1 + x + x^2$
x^5	$1 + x$
x^6	$x + x^2$
x^7	1

Table 3: Encoding Matrix G for $p(x) = 1 + x^2 + x^3$

Power of x	0	1	2	3	4	5	6
	1	0	1	1	0	0	0
G =	0	1	0	1	1	0	0
	0	0	1	0	1	1	0
	0	0	0	1	0	1	1

is divided by the generator polynomial $p(x)$. There is, however, a much better way of doing this. Since a row in the remainder table is the remainder of x^j, and the previous row is x^{j+1}, each row will be the remainder of the previous row after it is multiplied by x. This means, for example, if we are computing the remainder table for $p(x) = 1 + x^2 + x^3$, to obtain row i, we take the previous row and multiply by x. If there was an x^2 in the previous row, then we replace the resulting x^3 term by $1 + x^2$. This means that each entry in a given row is a sum (mod 2) of certain terms in the previous row, a formula we can easily set up in a spreadsheet.

4 Examples of Polynomial Codes

Suppose 1101101 was received. We add the remainders of the powers of x in this sequence using our monomial remainder table. Alternatively, we can divide $r(x) = 1 + x + x^3 + x^4 + x^6$ by $p(x) = 1 + x^2 + x^3$ and look at the remainder. In both cases, we get $1 + x + x^2$ which is the remainder of the monomial x^4. Thus, the 5th digit is in error. We correct it and the codeword is 1101001. Upon dividing 1101001 by 1011, we get 1111, the message.

We have now seen that polynomial codes are special cases of linear codes. From a generator polynomial $P(x)$, we can generate the encoding matrix with cyclic shifts, then we can transform this encoding matrix to $[I\,P]$ form through linear operations, and then compute the parity check matrix H for our linear code. Alternatively, we can divide our received sequence by $P(x)$, find the remainder and the associated monomial from the table of remainders. These two operations are exactly the same. The parity check matrix H is the remainder table. Multiplying by H is equivalent to computing the remainder of $r(x)$ by addition. Comparing the result to remainders of monomials and the rows of H is the same comparison. Homework problem: Convince yourself of this.

For a one-error-correcting polynomial code, we encode by multiplying the message $m(x)$ and our generator polynomial $p(x)$ to get a codeword $c(x) = m(x)p(x)$. For this to correct one error, we need that the degree of $c(x)$ is at most $2^d - 2$, as the remainders of x^0 and x^{2^d-1} are equal, and thus the code cannot distinguish between these two errors. The degree of $p(x)$ is d, and so the degree of $m(x)$ can be at most $2^d - 2 - d$. Since the number of coefficients of a polynomial is one more than its degree, these codes encode $2^d - d - 1$ message bits into $2^d - 1$ bits, and correct one error. These are exactly the same parameters as the Hamming codes we discovered by considering matrices in the previous lecture, and in fact, they are indeed exactly the same codes.

5 More on Finite Fields and Polynomials

In preparation for the next section on multiple error correcting codes, we will need a few more things about polynomials over finite fields.

When we computed polynomials $\pmod{p(x)}$, we have imposed the condition that $p(x) = 0$. We would like to know if higher powers of x also satisfy this condition, i.e., if $p(x^2) = 0$ or $p(x^3) = 0$, and so forth. The motivation will be apparent in the next section on BCH codes.

Note that if $p(x) = 0$, then $p(x^2) = 0$. Since the cross terms of $[p(x)]^2$ all disapper (2=0), we have $[p(x)]^2 = p(x^2) = 0$. Similarly, we can also show that x obeys all the equations that x^2 does.

If $p(x) = 0$, then we can find an equation which x^3 obeys by the following procedure.

- Form all the powers of x^{3j} for j up the degree of $p(x)$.
- Compute their remainders.
- Sum the remainders and eliminate all powers in those remainders.
- The equation left will be obeyed by x^3.

Similarily, we can find an equation obeyed by any power of x. Let's see this via an example.

Table 4: Remainder Table

Monomial	$p(x) = 1 + x + x^4$
1	1
x	x
x^2	x^2
x^3	x^3
x^4	$1 + x$
x^5	$x + x^2$
x^6	$x^2 + x^3$
x^7	$1 + x \quad +x^3$
x^8	$1 \quad + x^2$
x^9	$x \quad +x^3$
x^{10}	$1 + x + x^2$
x^{11}	$x + x^2 + x^3$
x^{12}	$1 + x + x^2 + x^3$
x^{13}	$1 \quad + x^2 + x^3$
x^{14}	$1 \quad +x^3$
x^{15}	1

Example 2: $p(x) = 1 + x + x^4$

Given $p(x) = 0$, we attempt to find $q(x) = 0$ such that $q(x^3) = 0$. For the first step, we have $\text{Rem}(x^6) = x^2 + x^3$ and $\text{Rem}(x^9) = x + x^3$ and $\text{Rem}(x^{12}) = 1 + x + x^2 + x^3$. For the next step, we eliminate x and x^2 to get $\text{Rem}(x^6 + x^9 + x^{12}) = 1 + x^3$. This implies $\text{Rem}(1 + x^3 + x^6 + x^9 + x^{12}) = 0$. Therefore, y such that $y = x^3$ obeys

$$1 + y + y^2 + y^3 + y^4 = 0 \tag{6}$$

Thus, $q(x) = 1 + x + x^2 + x^3 + x^4$.

Suppose we have $p(x)$ and wish to find a $w(x) = 0$ such that $w(x^5) = 0$. We have $\text{Rem}(x^5) = x + x^2$ and $\text{Rem}(x^{10}) = 1 + x + x^2$. So, $1 + x^5 + x^{10}$ has remainder 0. This implies that $z = x^5$ obeys the equation $1 + z + z^2 = 0$. Thus, $w(x) = 1 + x + x^2$.

Since x^2 always obeys the same equation as x, then x^4 and x^8 must also obey the same equation. Likewise, x^3, x^6, x^{12}, $x^{24} = x^9$ all must obey the same equation, Similarily for the other powers. We can find all the equations and all the powers which obey the equations in the table below.

Table 5: $p(x) = 0$ and $p(x^j) = 0$

power	Equation for $p(x)$ satisfied power
1, 2, 4, 8	$1 + x + x^4 = 0$
3, 6, 9, 12	$1 + x^2 + x^3 + x^4 = 0$
5, 10	$1 + x + x^2 = 0$
7, 11, 13, 14	$1 + x^3 + x^4 = 0$
0	$1 + x = 0$

There are two fundamental facts about the linearity of remainders that are very useful in computation. We can find remainders by matrix multiplication and also by long division. The two facts below say that we can find remainders by adding the remainders of the monomials and we can find remainders of products by multiplying their remainders.

$$\mathrm{Rem}[\,a(x) + b(x)\,] = \mathrm{Rem}[\,a(x)\,] + \mathrm{Rem}[\,b(x)\,] \tag{7}$$
$$\mathrm{Rem}[\,a(x)\,b(x)\,] = \mathrm{Rem}[\,a(x)\,]\,\mathrm{Rem}[\,b(x)\,] \tag{8}$$

In conclusion, we have seen that if the generator polynomial is primitive, then the code generated will be a perfect single error correcting code of length $n = 2^r - 1$. Each codeword is an n bit length string corresponding to polynomials divisible by $p(x)$. To correct one error, represented as a monomial, we compare the remainder of the received polynomial $r(x)$ upon dividing by $p(x)$ and compare with the remainders of the monomials in our remainder table.