CS 70 Discrete Mathematics and Probability Theory
Spring 2019 Course Notes **Note 9**

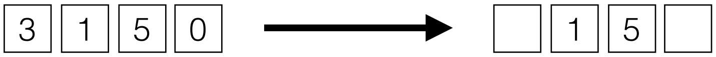
Error Correcting Codes

In this note, we will discuss the problem of transmitting messages across an unreliable communication channel. The channel may cause some parts of the message ("packets") to be lost, or dropped; or, more seriously, it may cause some packets to be corrupted. We will learn how to "encode" the message by introducing redundancy into it in order to protect against both of these types of errors. Such an encoding scheme is known as an "error correcting code." Error correcting codes are a major object of study in mathematics, computer science and electrical engineering; they belong to a field known as "Information Theory." In addition to the beautiful theory underlying them (which we will glimpse in this note), they are of great practical importance: every time you use your cellphone, satellite TV, DSP, cable modem, disk drive, CD-ROM, DVD player etc., or send and receive data over the internet, you are using error correcting codes to ensure that information is transmitted reliably.

There are, very roughly speaking, (at least) two distinct flavors of error correcting codes: algebraic codes, which are based on polynomials over finite fields, and combinatorial codes, which are based on graph theory. In this note we will focus on algebraic codes, and in particular on so-called Reed-Solomon codes (named after two of their inventors). In doing so, we will be making essential use of the properties we learned about polynomials in the last lecture.

Erasure Errors

We will consider two situations in which we wish to transmit information on an unreliable channel. The first is exemplified by the Internet, where the information (say a file) is broken up into packets, and the unreliability is manifest in the fact that some of the packets are lost during transmission, as shown below:



We refer to such errors as *erasure errors*. Suppose that the message consists of n packets and suppose that at most k packets are lost during transmission. We will show how to encode the initial message consisting of n packets into a redundant encoding consisting of $n+k$ packets such that the recipient can reconstruct the message from *any* n received packets. Note that in this setting the packets are labeled with headers, and thus the recipient knows exactly which packets were dropped during transmission.

We can assume without loss of generality that the content of each packet is a number modulo q, where q is a prime. For example, the content of the packet might be a 32-bit string and can therefore be regarded as a number between 0 and $2^{32}-1$; then we could choose q to be any prime larger than 2^{32}. The properties of polynomials over $GF(q)$ (i.e., with coefficients and values reduced modulo q) are perfectly suited to solve this problem and are the backbone of this error-correcting scheme.

To see this, let us denote the message to be sent by $m_1,\dots,m_n$, where each m_i is a number in $GF(q)$, and make the following crucial observations:

1. There is a unique polynomial $P(x)$ of degree $n-1$ such that $P(i) = m_i$ for $1 \leq i \leq n$ (i.e., $P(x)$ contains all of the information about the message, and evaluating $P(i)$ gives the intended contents of the i-th packet).

2. The message to be sent is now $m_1 = P(1), \ldots, m_n = P(n)$. We can generate additional packets by evaluating $P(x)$ at points $n+j$. (Recall that our transmitted message should be redundant, i.e., it should contain more packets than the original message to account for the lost packets.) Thus the transmitted message is $c_1 = P(1), c_2 = P(2), \ldots, c_{n+k} = P(n+k)$. Since we are working modulo q, we must make sure that $n+k \leq q$, but this condition does not impose a serious constraint since q is assumed to be very large.

3. We can uniquely reconstruct $P(x)$ from its values at *any* n distinct points, since it has degree $n-1$. This means that $P(x)$ can be reconstructed from *any* n of the transmitted packets (not just the original n packets). Once we have reconstructed the polynomial P, we can evaluate $P(x)$ at $x = 1, \ldots, n$ to recover the original message $m_1, \ldots, m_n$.

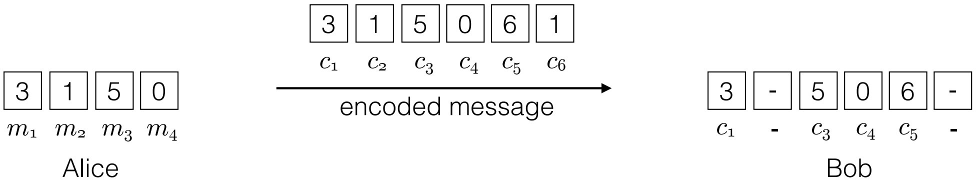
Example

Suppose Alice wants to send Bob a message of $n = 4$ packets and she wants to guard against $k = 2$ lost packets. Then, assuming the packets can be coded up as integers between 0 and 6, Alice can work over $GF(7)$ (since $7 \geq n+k = 6$; of course, in real applications, we would be working over a much larger field!). Suppose the message that Alice wants to send to Bob is $m_1 = 3$, $m_2 = 1$, $m_3 = 5$, and $m_4 = 0$. The unique polynomial of degree $n-1 = 3$ described by these 4 points is $P(x) = x^3 + 4x^2 + 5$.

Exercise. We derived this polynomial using Lagrange interpolation mod 7. Check this derivation, and verify also that indeed $P(i) = m_i$ for $1 \leq i \leq 4$.

Since $k = 2$, Alice must evaluate $P(x)$ at 2 extra points: $P(5) = 6$ and $P(6) = 1$. Now, Alice can transmit the encoded message which consists of $n+k = 6$ packets, where $c_j = P(j)$ for $1 \leq j \leq 6$. So Alice will send $c_1 = P(1) = 3$, $c_2 = P(2) = 1$, $c_3 = P(3) = 5$, $c_4 = P(4) = 0$, $c_5 = P(5) = 6$, and $c_6 = P(6) = 1$.

Now suppose packets 2 and 6 are dropped, in which case we have the following situation:



From the values that Bob received (3, 5, 0, and 6), he uses Lagrange interpolation and computes the following basis polynomials (where everything should be interpreted mod 7):

$$\begin{aligned}
\Delta_1(x) &= \frac{(x-3)(x-4)(x-5)}{-24} \equiv 2(x-3)(x-4)(x-5) \pmod 7 \\
\Delta_3(x) &= \frac{(x-1)(x-4)(x-5)}{4} \equiv 2(x-1)(x-4)(x-5) \pmod 7 \\
\Delta_4(x) &= \frac{(x-1)(x-3)(x-5)}{-3} \equiv 2(x-1)(x-3)(x-5) \pmod 7 \\
\Delta_5(x) &= \frac{(x-1)(x-3)(x-4)}{8} \equiv (x-1)(x-3)(x-4) \pmod 7.
\end{aligned}$$

(Note that we have used the fact here that the inverses of -24, 4, -3 and 8 $\pmod 7$ are 2, 2, 2 and 1 respectively.) He then reconstructs the polynomial $P(x) = 3 \cdot \Delta_1(x) + 5 \cdot \Delta_3(x) + 0 \cdot \Delta_4(x) + 6 \cdot \Delta_5(x) =$

$x^3+4x^2+5 \pmod 7$. Bob then evaluates $m_2 = P(2) = 1$, which is the packet that was lost from the original message. More generally, no matter which two packets were dropped, following exactly the same method Bob can always reconstruct $P(x)$ and thus the original message.

Exercise. Check Bob's calculation above, and verify that he really does reconstruct the correct polynomial, as claimed. Remember that all arithmetic must be done mod 7.

Let us consider what would happen if Alice sent one fewer packet. If Alice only sent c_j for $1 \le j \le n+k-1$, then with k erasures, Bob would only receive c_j for $n-1$ distinct values j. Thus, Bob would not be able to reconstruct $P(x)$ (since there are q polynomials of degree at most $n-1$ that agree with the $n-1$ packets which Bob received)! This error-correcting scheme is therefore optimal: it can recover the n characters of the transmitted message from any n received characters, but recovery from any smaller number of characters is impossible.

Polynomial Interpolation Revisited

Let us take a brief digression to discuss another method of polynomial interpolation (different from Lagrange interpolation discussed in the previous note) which will be useful in handling general errors. Again, the goal of the algorithm will be to take as input $d+1$ pairs $(x_1,y_1),\cdots,(x_{d+1},y_{d+1})$, and output the polynomial $p(x) = a_dx^d+\cdots+a_1x+a_0$ such that $p(x_i) = y_i$ for $i = 1$ to $d+1$.

The first step of the algorithm is to write a system of $d+1$ linear equations in $d+1$ variables: the variables are the coefficients of the polynomial, $a_0,\dots,a_d$. Each equation is obtained by fixing x to be one of $d+1$ values: $x_1,\cdots,x_{d+1}$. Note that in $p(x)$, x is a variable and $a_0,\dots,a_d$ are fixed constants. In the equations below, these roles are swapped: x_i is a fixed constant and $a_0,\dots,a_d$ are variables. For example, the i-th equation is the result of fixing x to be x_i, and saying that the value of the polynomial is y_i: $a_dx_i^d + a_{d-1}x_i^{d-1} + \ldots + a_0 = y_i$.

Now solving these equations gives the coefficients of the polynomial $p(x)$. For example, suppose we are given the three pairs $(-1,2)$, $(0,1)$, and $(2,5)$; our goal is to construct the degree-2 polynomial $p(x)$ which goes through these points. The first equation says $a_2(-1)^2+a_1(-1)+a_0 = 2$. Simplifying, we get $a_2 - a_1 + a_0 = 2$. Similarly, the second equation says $a_2(0)^2+a_1(0)+a_0 = 1$, or $a_0 = 1$. And the third equation says $a_2(2)^2+a_1(2)+a_0 = 5$ So we get the following system of equations:

$$\begin{aligned} a_2 - a_1 + a_0 &= 2 \\ a_0 &= 1 \\ 4a_2 + 2a_1 + a_0 &= 5 \end{aligned}$$

Substituting for a_0 and multiplying the first equation by 2 we get:

$$\begin{aligned} 2a_2 - 2a_1 &= 2 \\ 4a_2 + 2a_1 &= 4 \end{aligned}$$

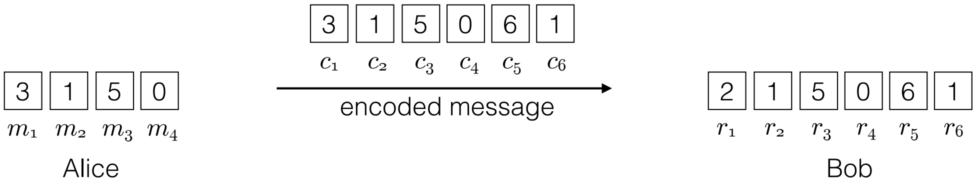
Then, adding the two equations we find that $6a_2 = 6$, so $a_2 = 1$, and plugging back in we find that $a_1 = 0$. Thus, we have determined the polynomial $p(x) = x^2+1$. To justify this method more carefully, we must show that the equations always have a solution and that it is unique. This involves showing that a certain determinant is non-zero, which we will leave as an (optional) exercise; for now we will assume that there is always a unique solution.

General Errors

Let us now return to our main topic of error correction, and consider a much more challenging scenario. Suppose that Alice wishes to communicate with Bob over a noisy channel (say, via a modem). Her message is $m_1,\dots,m_n$, where we may think of the m_i's as characters (either bytes or characters in the English alphabet). The problem now is that some of the characters are *corrupted* during transmission due to channel noise. Thus Bob receives exactly as many characters as Alice transmits, but k of them are corrupted, and Bob has no idea which k these are! Recovering from such general errors is much more challenging than recovering from erasure errors, though once again polynomials hold the key. As we shall see, Alice can still guard against k general errors, at the expense of transmitting only $2k$ additional characters (twice as many as in the erasure case we saw above).

We will again think of each character as a number modulo q for some prime q. (For the English alphabet, q is some prime larger than 26, say $q = 29$.) As before, we can describe the message by a polynomial $P(x)$ of degree $n-1$ over $GF(q)$, such that $P(1) = m_1$, ..., $P(n) = m_n$. As before, to cope with the errors Alice will transmit additional characters obtained by evaluating $P(x)$ at additional points. As mentioned above, in order to guard against k general errors, Alice must transmit $2k$ additional characters: thus the encoded message is $c_1,\dots,c_{n+2k}$ where $c_j = P(j)$ for $1 \le j \le n+2k$, and $n+k$ of these characters that Bob receives are uncorrupted. As before, we must put the mild constraint on q that it be large enough so that $q \ge n+2k$.

For example, if Alice wishes to send $n = 4$ characters to Bob via a modem in which $k = 1$ of the characters is corrupted, she must redundantly send an encoded message consisting of 6 characters. Suppose she wants to transmit the same message (3051) as in our erasure example above, and that c_1 is corrupted from 3 to 2. This scenario can be visualized in the following figure:


3
1
5
0
6
1
c_1 c_2 c_3 c_4 c_5 c_6
3
1
5
0
m_1 m_2 m_3 m_4
Alice
encoded message
2
1
5
0
6
1
r_1 r_2 r_3 r_4 r_5 r_6
Bob

From Bob's viewpoint, the problem of reconstructing Alice's message is the same as reconstructing the polynomial $P(x)$ from the $n+2k$ received characters $r_1, r_2, \dots, r_{n+2k}$. In other words, Bob is given $n+2k$ values, $r_1, r_2, \dots, r_{n+2k}$ modulo q, with the promise that there is a polynomial $P(x)$ of degree $n-1$ over $GF(q)$ such that $P(i) = r_i$ for $n+k$ distinct values of i between 1 and $n+2k$. Bob must reconstruct $P(x)$ from this data (in the above example, $n+k = 5$, and $r_2 = P(2) = 1$, $r_3 = P(3) = 5$, $r_4 = P(4) = 0$, $r_5 = P(5) = 6$, and $r_6 = P(6) = 1$). Note, however, that Bob does not know *which* of the $n+k$ values are correct!

Does Bob even have sufficient information to reconstruct $P(x)$? Our first observation shows that the answer is yes: for any given subset of $n+k$ values of i between 1 and $n+2k$, there is a unique polynomial $P(x)$ such that $P(i) = r_i$ at these values of i. To see this, suppose that $P'(x)$ is any polynomial of degree $n-1$ that goes through these $n+k$ points. Then among these $n+k$ points there are at most k errors, and therefore on at least n of the points we must have $P'(i) = P(i)$. But, as we saw in the previous note (Property 2), a polynomial of degree $n-1$ is uniquely defined by its values at n points, and therefore $P'(x)$ and $P(x)$ must be the same polynomial.

But how can Bob efficiently find such a polynomial? The issue at hand is the locations of the k errors. Let $e_1,\dots,e_k$ be the k locations at which errors occurred (so that $P(i) = r_i$ for all $i \notin \{e_1,\dots,e_k\}$).

Now Bob could try to guess where the k errors lie, but this would take too long (it would take exponential time, in fact). Instead, Bob will employ a clever trick based on the following definition. Consider the so-called *error-locator polynomial*

$$E(x) = (x-e_1)(x-e_2)\cdots(x-e_k).$$

Note that $E(x)$ is a polynomial of degree k (since x appears k times). Note also that Bob does not know this polynomial explicitly, because he does not know the positions e_i of the errors. However, he will use the polynomial symbolically, and will eventually compute the values e_i that appear in it!

Let us make a simple but crucial observation about this polynomial:

$$P(i)E(i) = r_iE(i) \qquad \text{for } 1 \le i \le n+2k. \tag{1}$$

To see this, note that it holds at points i at which no error occurred since at those points $P(i) = r_i$; and it is trivially true at points i at which an error occurred since then $E(i) = 0$. This observation forms the basis of a very clever algorithm invented by Berlekamp and Welch. Looking more closely at the equalities in (1), we will show that they in fact correspond to $n+2k$ linear equations in $n+2k$ unknowns, from which the locations of the errors and coefficients of $P(x)$ can be easily deduced (in analogous fashion to the interpolation scheme we saw in the previous section).

Now define $Q(x) := P(x)E(x)$, which is a polynomial of degree $n+k-1$, and is therefore described by $n+k$ coefficients $a_0, a_1, \ldots, a_{n+k-1}$. The error-locator polynomial $E(x) = (x-e_1)\cdots(x-e_k)$ has degree k and is described by $k+1$ coefficients $b_0, b_1, \ldots, b_k$ but the leading coefficient (the coefficient b_k of x^k) is always 1. So we can write:

$$\begin{aligned} Q(x) &= a_{n+k-1}x^{n+k-1} + \cdots + a_1x + a_0 \\ E(x) &= x^k + b_{k-1}x^{k-1} + \cdots + b_1x + b_0 \end{aligned}$$

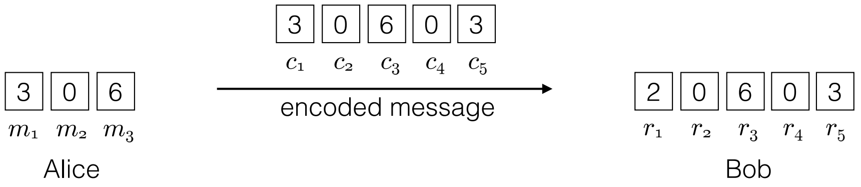
Once we fix a value i for x, the received value r_i is fixed. Also, $Q(i)$ is now a linear function of the $n+k$ coefficients $a_{n+k-1}\ldots a_0$, and $E(i)$ is a linear function of the k coefficients $b_{k-1}\ldots b_0$. Therefore the equation $Q(i) = r_iE(i)$ (which just comes from rewriting (1) using $Q(x) = P(x)E(x)$) is a linear equation in the $n+2k$ unknowns $a_{n+k-1}, \ldots, a_0$ and $b_{k-1}, \ldots, b_0$. We thus have $n+2k$ linear equations, one for each value of i, and $n+2k$ unknowns. We can solve these equations and get $E(x)$ and $Q(x)$. We can then compute the ratio $\frac{Q(x)}{E(x)}$ to obtain $P(x)$. This is best illustrated by an example.

Example. Suppose we are working over $GF(7)$ and Alice wants to send Bob the $n = 3$ characters "3," "0," and "6" over a modem. Turning to the analogy of the English alphabet, this is equivalent to using only the first 7 letters of the alphabet, where $a = 0, \ldots, g = 6$. So the message which Alice wishes for Bob to receive is "dag". Then Alice interpolates (e.g., using Lagrange: exercise!) to find the polynomial

$$P(x) = x^2 + x + 1,$$

which is the unique polynomial of degree 2 such that $P(1) = 3$, $P(2) = 0$, and $P(3) = 6$.

Suppose Alice knows that $k = 1$ character will be corrupted; then she needs to transmit the $n+2k = 5$ characters $P(1) = 3$, $P(2) = 0$, $P(3) = 6$, $P(4) = 0$, and $P(5) = 3$ to Bob. Suppose $P(1)$ is corrupted, and Bob receives the character 2 instead of 3 (i.e., Alice sends the encoded message "dagad" but Bob instead receives "cagad"). Summarizing, we have the following situation:



Let $E(x) = (x - e_1) = x + b_0$ be the error-locator polynomial—remember, Bob doesn't know what $e_1 = -b_0$ is yet since he doesn't know where the error occurred—and let $Q(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ (where again the coefficients a_i are unknown). Now Bob just substitutes $i = 1$, $i = 2$, ..., $i = 5$ into the equations $Q(i) = r_iE(i)$ from (1) above and simplifies to get five linear equations in five unknowns (recall that we are working modulo 7 and that r_i is the value Bob received for the i-th character):

$$\begin{aligned}
a_3 + a_2 + a_1 + a_0 + 5b_0 &= 2\\
a_3 + 4a_2 + 2a_1 + a_0 &= 0\\
6a_3 + 2a_2 + 3a_1 + a_0 + b_0 &= 4\\
a_3 + 2a_2 + 4a_1 + a_0 &= 0\\
6a_3 + 4a_2 + 5a_1 + a_0 + 4b_0 &= 1
\end{aligned}$$

Bob then solves this linear system and finds that $a_3 = 1$, $a_2 = 0$, $a_1 = 0$, $a_0 = 6$ and $b_0 = 6$ (all mod 7). (As a check, this implies that $E(x) = x + 6 = x - 1$, so the location of the error is position $e_1 = 1$, which is correct since the first character was corrupted from a "d" to a "c".) This gives him the polynomials $Q(x) = x^3 + 6$ and $E(x) = x - 1$. He can then find $P(x)$ by computing the quotient $P(x) = \frac{Q(x)}{E(x)} = \frac{x^3+6}{x-1} = x^2 + x + 1$. Bob notices that the first character was corrupted (since $e_1 = 1$), so now that he has $P(x)$, he just computes $P(1) = 3 =$ "d" and obtains the original, uncorrupted message "dag".

Exercise. Verify the derivation of the above system of equations from the equalities $Q(i) = r_iE(i)$ for $i = 1,2,3,4,5$. Also, solve the equations and check that your solution agrees with the one above. Remember that all the arithmetic should be done mod 7.

Exercise. Redo the example above for the case where the second character is corrupted from a "0" to a "5", and all other characters are uncorrupted.

Finer Points

Two points need further discussion. How do we know that the $n + 2k$ equations are *consistent*? What if they have no solution? This is simple. The equations must be consistent since $Q(x) = P(x)E(x)$ together with the error locator polynomial $E(x)$ gives a solution.

A more interesting question is this: how do we know that the $n + 2k$ equations are *independent*, i.e., how do we know that there aren't other spurious solutions in addition to the real solution that we are looking for? Put more mathematically, suppose that the solution we construct is $Q'(x), E'(x)$; how do we know that this solution satisfies the property that $E'(x)$ divides $Q'(x)$ and that $\frac{Q'(x)}{E'(x)} = \frac{Q(x)}{E(x)} = P(x)$?

To see that this is true, we note first that, based on our method for calculating $Q'(x), E'(x)$, we know that $Q'(i) = r_iE'(i)$ for $1 \le i \le n + 2k$; and of course we also have, by definition, $Q(i) = r_iE(i)$ for the same values of i. Multiplying the first of these equations by $E(i)$ and the second by $E'(i)$, we get

$$Q'(i)E(i) = Q(i)E'(i) \qquad \text{for } 1 \le i \le n + 2k, \tag{2}$$

since both sides are equal to $r_i E(i)E'(i)$. Equation (2) tells us that the two polynomials $Q(x)E'(x)$ and $Q'(x)E(x)$ are equal at $n+2k$ points. But these two polynomials both have degree $n+2k-1$, so they are completely determined by their values at $n+2k$ points. Therefore, since they agree at $n+2k$ points, they must be the same polynomial, i.e., $Q(x)E'(x) = Q'(x)E(x)$ for all x.[1] Now we may divide through by the polynomial $E(x)E'(x)$ (which by construction is not the zero polynomial) to obtain $\frac{Q'(x)}{E'(x)} = \frac{Q(x)}{E(x)} = P(x)$, which is what we wanted. Hence we can be sure that any solution we find is correct.

Exercise. Is the solution $Q'(x), E'(x)$ always unique? The above analysis tells us that the ratio must always satisfy $\frac{Q'(x)}{E'(x)} = \frac{Q(x)}{E(x)} = P(x)$, but does not guarantee that $Q'(x) = Q(x)$ and $E'(x) = E(x)$. Hint: What happens in our example above when in fact none of the characters is corrupted (although Alice still assumes that $k = 1$ character may be corrupted)? Try writing out and solving the equations in this case.

---

[1] Note that this is a much stronger statement than equation (2). Equation (2) says that the *values* of the polynomials agree at certain points i; this statement says that the two polynomials are equal everywhere, i.e., they are the same polynomial.