Jingjia Chen, Michael Psenka and Tarang Srivastava

1 Berlekamp-Welch Warm Up

Let $P(i)$, a polynomial applied to the input i, be the original encoded polynomial before sent, and let r_i be the received info for the input i which may or may not be corrupted.

(a) If you want to send a length-n message, what should the degree of $P(x)$ be? Why?

(b) When does $r_i = P(i)$? When does r_i not equal $P(i)$?

(c) If there are at most k erasure errors, how many packets should you send? If there are at most k general errors, how many packets should you send? (We will see the reason for this later.) Now we will only consider general errors.

(d) What do the roots of the error polynomial $E(x)$ represent? Does the receiver know the roots of $E(x)$? If there are at most k errors, what is the maximum degree of $E(x)$? Using the information about the degree of $P(x)$ and $E(x)$, what is the degree of $Q(x) = P(x)E(x)$?

(e) Why is the equation $Q(i) = P(i)E(i) = r_iE(i)$ always true? (Consider what happens when $P(i) = r_i$, and what happens when $P(i)$ does not equal r_i.)

(f) In the polynomials $Q(x)$ and $E(x)$, how many total unknown coefficients are there? (These are the variables you must solve for. Think about the degree of the polynomials.) When you receive packets, how many equations do you have? Do you have enough equations to solve for all of the unknowns? (Think about the answer to the earlier question - does it make sense now why we send as many packets as we do?)

(g) If you have $Q(x)$ and $E(x)$, how does one recover $P(x)$? If you know $P(x)$, how can you recover the original message?

Solution:

(a) P has degree $n-1$ since n points would determine a degree $n-1$ polynomial.

(b) $r_i = P(i)$ when the received packet is correct. r_i does not equal $P(i)$ the received packet is corrupted.

(c) We send $n+k$ packets when we have k erasures and $n+2k$ packets for k general errors.

(d) The roots of error polynomial $E(x)$ represent the locations of corrupted packets. The receiver does not know the roots of $E(x)$. $E(x)$ is a polynomial that the receiver needs to compute in order to obtain $P(x)$. If there are at most k errors, then the maximum degree of $E(x)$ is k. The maximum degree of Q is $(n-1)+(k)=n+k-1$ since the degree of P is $n-1$ and the degree of E is at most k.

(e) If there is no error at point i, $P(i)=r_i$ and then multiplying each side by $E(i)$ gives $P(i)E(i)=r_iE(i)$. If there is an error at point i, then $E(i)=0$, which means $P(i)E(i)=r_iE(i)=0$.

(f) The maximum degree of $Q(x)$ is $n+k-1$, so the number of unknowns is $n+k$. The maximum degree of $E(x)$ is k, which would mean there would be $k+1$ unknowns. However, we know that the coefficient of x^k is 1 in $E(x)$, so the number of unknowns is k.

The total number of unknowns is $(n+k)+(k)=n+2k$

There are $n+2k$ equations, which is enough to solve for $n+2k$ unknowns.

(g) We can compute $P(x)$ using the equation: $P(x)=Q(x)/E(x)$. To recover the message, we compute $P(i)$ for $1 \le i \le n$.

2 Berlekamp-Welch Algorithm

In this question we will send the message $(m_0, m_1, m_2) = (1, 1, 4)$ of length $n=3$. We will use an error-correcting code for $k=1$ general error, doing arithmetic over GF(5).

(a) Construct a polynomial $P(x) \pmod 5$ of degree at most 2, so that

$$P(0)=1, \qquad P(1)=1, \qquad P(2)=4.$$

What is the message $(c_0, c_1, c_2, c_3, c_4)$ that is sent?

(b) Suppose the message is corrupted by changing c_0 to 0. Set up the system of linear equations in the Berlekamp-Welch algorithm to find $Q(x)$ and $E(x)$.

(c) Assume that after solving the equations in part (b) we get $Q(x)=4x^3+x^2+x$ and $E(x)=x$. Show how to recover the original message from Q and E.

Solution:

(a) We use Lagrange interpolation to construct the unique quadratic polynomial $P(x)$ such that $P(0)=m_0=1, P(1)=m_1=1, P(2)=m_2=4$. Doing all arithmetic over GF(5), so that i.e.

$2^{-1} = 3 \pmod 5$,

$$\Delta_0(x) = \frac{(x-1)(x-2)}{(0-1)(0-2)} = \frac{x^2-3x+2}{2} \equiv 3(x^2-3x+2) \pmod 5$$
$$\Delta_1(x) = \frac{(x-0)(x-2)}{(1-0)(1-2)} = \frac{x^2-2x}{-1} \equiv 4(x^2-2x) \pmod 5$$
$$\Delta_2(x) = \frac{(x-0)(x-1)}{(2-0)(2-1)} = \frac{x^2-x}{2} \equiv 3(x^2-x) \pmod 5$$
$$\begin{aligned} P(x) &= m_0\Delta_0(x) + m_1\Delta_1(x) + m_2\Delta_2(x) \\ &= 1\Delta_0(x) + 1\Delta_1(x) + 4\Delta_2(x) \\ &\equiv 4x^2 + x + 1 \pmod 5 \end{aligned}$$

For the final message we need to add 2 redundant points of P. Since 3 and 4 are the only points in GF(5) that we have not used yet, we compute $P(3) = 0, P(4) = 4$, and so our message is $(1,1,4,0,4)$.

(b) The message received is $(c'_0, c'_1, c'_2, c'_3, c'_4) = (0,1,4,0,4)$. Let $R(x)$ be the function such $R(i) = c'_i$ for $0 \le i < 5$. Let $E(x) = x + b_0$ be the error-locator polynomial, and $Q(x) = P(x)E(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. Since $Q(i) = P(i)E(i) = R(i)E(i)$ for $1 \le i < 5$, we have the following equalities $\pmod 5$

$$\begin{aligned} Q(0) &= 0E(0) \\ Q(1) &= 1E(1) \\ Q(2) &= 4E(2) \\ Q(3) &= 0E(3) \\ Q(4) &= 4E(4) \end{aligned}$$

which can be rewritten as a system of linear equations

$$\begin{array}{ccccccccccc} & & & & & & a_0 & & & = & 0 \\ a_3 & + & a_2 & + & a_1 & + & a_0 & - & b_0 & = & 1 \\ 8a_3 & + & 4a_2 & + & 2a_1 & + & a_0 & - & 4b_0 & = & 8 \\ 27a_3 & + & 9a_2 & + & 3a_1 & + & a_0 & & & = & 0 \\ 64a_3 & + & 16a_2 & + & 4a_1 & + & a_0 & - & 4b_0 & = & 1 \end{array}.$$

(c) From the solution, we know

$$\begin{aligned} Q(x) &= 4x^3 + x^2 + x, \\ E(x) &= x + b_0 = x. \end{aligned}$$

Since $Q(x) = P(x)E(x)$, the recipient can compute $P(x) = Q(x)/E(x) = 4x^2 + x + 1$, the polynomial $P(x)$ from part (a) used by the sender. The error locating polynomial $E(x)$ is degree one, so there is only one error, and as $E(x) = x = x - 0$, the corrupted bit was the first one. To correct this error we evaluate $P(0) = 1$ and combine this with the two uncorrupted bits m_1, m_2, to get the original message

$$(m_0, m_1, m_2) = (1,1,4).$$

3 Berlekamp-Welch Algorithm with Fewer Errors

In class we derived how the Berlekamp-Welch algorithm can be used to correct k general errors, given $n+2k$ points transmitted. In real life, it is usually difficult to determine the number of errors that will occur. What if we have less than k errors? This is a follow up to the exercise posed in the notes.

Suppose Alice wants to send 1 message to Bob and wants to guard against 1 general error. She decides to encode the message with $P(x) = 4$ (on GF(7)) such that $P(0) = 4$ is the message she want to send. She then sends $P(0), P(1), P(2) = (4,4,4)$ to Bob.

(a) Suppose Bob receives the message $(4,5,4)$. Without performing Gaussian elimination explicitly, find $E(x)$ and $Q(x)$.

(b) Now, suppose there were no general errors and Bob receives the original message $(4,4,4)$. Show that the $Q(x), E(x)$ that you found in part (a) still satisfies $Q(i) = r_i E(i)$ for all $i = 0,1,2$.

(c) Verify that $E(x) = x$, $Q(x) = 4x$ is another possible set of polynomials that satisfies $Q(i) = r_i E(i)$ for all $i = 0,1,2$.

(d) Suppose you're actually trying to decode the received message $(4,4,4)$. Based on what you showed in the previous two parts, what will happen during row reduction when you try to solve for the unknowns?

(e) Prove that in general, no matter what the solution of $Q(x)$ and $E(x)$ are though, the recovered $P(x)$ will always be the same.

Solution:

(a) $E(x) = x-1$ and $Q(x) = P(x)E(x) = 4x-4$.

(b) This is true because there were no errors, so $P(i) = r_i$ for $i = 0,1,2$.

(c) Since $Q(x) = P(x)E(x)$ and $P(i) = r_i$ for $i = 0,1,2$, we must have $Q(i) = r_i E(i)$ for all $i = 0,1,2$.

(d) There are multiple solutions to the system of equations.

(e) Suppose we got two solutions $Q'(x), E'(x)$ and $Q(x), E(x)$. Since they are both solutions, by definition, we have $Q'(i) = r_i E'(i)$ and $Q(i) = r_i E(i)$ for $1 \le i \le n+2k$. Therefore, $Q'(i)E(i) = Q(i)E'(i) = r_i E(i)E'(i)$. However, $Q'(x)E(x) - Q(x)E'(x)$ is a degree $n+2k-1$ polynomial, which is 0 at $n+2k$ points. Thus, $Q'(x)E(x) = Q(x)E'(x)$ for all x, so we arrive at

$$\frac{Q'(x)}{E'(x)} = \frac{Q(x)}{E(x)}.$$

This proves that the final solution for $P(x)$ is the same.