CS 70 Discrete Mathematics and Probability Theory
Spring 2017 Rao

DIS 5b

1 Berlekamp-Welch Warm Up

(a) When does $r_i = P(i)$?

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

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

(d) If there are at most k erasure errors, how many packets should you send?

(e) 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.

(f) What do the roots of the error polynomial $E(x)$ tell you? Does the receiver know the roots of $E(x)$?

(g) If there are at most k errors, what is the maximum degree of $E(x)$?

(h) Using the information about the degree of $P(x)$ and $E(x)$, what can you conclude about the degree of $Q(x) = P(x)E(x)$?

(i) 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.)

(j) 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.)

(k) 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?)

(l) If you have $Q(x)$ and $E(x)$, how does one recover $P(x)$?

(m) If you know $P(x)$, how can you recover the original message?

Solution:

(a) The received packet is correct.

(b) The received packet is corrupted.

(c) $n-1$. n points determine a degree $n-1$ polynomial.

(d) $n+k$

(e) $n+2k$

(f) The locations of corrupted packets. No.

(g) k

(h) $(n-1)+(k) = n+k-1$

(i) If $P(i) = r_i$, then $P(i)E(i) = r_iE(i)$. If $P(i) \neq r_i$, then $E(i) = 0$.

(j) $(n+k-1+1)+(k) = n+2k$ unknowns.

(k) $n+2k$. Yes.

(l) $P(x) = Q(x)/E(x)$

(m) Compute $P(i)$ for $1 \leq i \leq n$.

2 Berlekamp-Welch for General Errors

Suppose that Hector wants to send you a length $n = 3$ message, m_0, m_1, m_2, with the possibility for $k = 1$ error. For all parts of this problem, we will work mod 11, so we can encode 11 letters as shown below:

A	B	C	D	E	F	G	H	I	J	K
0	1	2	3	4	5	6	7	8	9	10

Hector encodes the message by finding the degree ≤ 2 polynomial $P(x)$ that passes through $(0, m_0)$, $(1, m_1)$, and $(2, m_2)$, and then sends you the five packets $P(0), P(1), P(2), P(3), P(4)$ over a noisy channel. The message you receive is

$$\text{DHACK} \Rightarrow 3, 7, 0, 2, 10 = r_0, r_1, r_2, r_3, r_4$$

which could have up to 1 error.

(a) First, let's locate the error, using an error-locating polynomial $E(x)$. Let $Q(x) = P(x)E(x)$. Recall that

$$Q(i) = P(i)E(i) = r_iE(i), \quad \text{for} \quad 0 \leq i < n+2k.$$

What is the degree of $E(x)$? What is the degree of $Q(x)$? Using the relation above, write out the form of $E(x)$ and $Q(x)$ in terms of the unknown coefficients, and then a system of equations to find both these polynomials.

(b) Solve for $Q(x)$ and $E(x)$. Where is the error located?

(c) Finally, what is $P(x)$? Use $P(x)$ to determine the original message that Hector wanted to send.

Hint: The message refers to a US federal agency.

Solution:

(a) The degree of $E(x)$ will be 1, since there is at most 1 error. The degree of $Q(x)$ will be 3, since $P(x)$ is of degree 2. $E(x)$ will have the form $E(x) = x + e$, and $Q(x)$ will have the form $Q(x) = ax^3 + bx^2 + cx + d$. We can write out a system of equations to solve for these 5 variables:

$$\begin{aligned} d &= 3(0+e) \\ a+b+c+d &= 7(1+e) \\ 8a+4b+2c+d &= 0(2+e) \\ 27a+9b+3c+d &= 2(3+e) \\ 64a+16b+4c+d &= 10(4+e) \end{aligned}$$

Since we are working mod 11, this is equivalent to:

$$\begin{aligned} d &= 3e \\ a+b+c+d &= 7+7e \\ 8a+4b+2c+d &= 0 \\ 5a+9b+3c+d &= 6+2e \\ 9a+5b+4c+d &= 7+10e \end{aligned}$$

(b) Solving this system of linear equations we get

$$Q(x) = 3x^3 + 6x^2 + 5x + 8.$$

Plugging this into the first equation (for example), we see that:

$$d = 8 = 3e \quad \Rightarrow \quad e = 8 \cdot 4 = 32 \equiv 10 \mod 11$$

This means that

$$E(x) = x + 10 \equiv x - 1 \mod 11.$$

Therefore, the error occurred at $x = 1$ (so the second number sent in this case).

(c) Using polynomial division, we divide $Q(x) = 3x^3 + 6x^2 + 5x + 8$ by $E(x) = x - 1$:

$$P(x) = 3x^2 + 9x + 3$$

Then, $P(1) = 3 + 9 + 3 = 15 \equiv 4 \mod 11$. This means that our original message was

$$3, 4, 0 \quad \Rightarrow \quad \text{DEA}.$$

Note: In Season 4 of Breaking Bad, Hector Salamanca (who cannot speak), uses a bell to spell out "DEA" (Drug Enforcement Agency).

3 List Decoding

(a) Consider an n character message encoded into m characters over the field GF(p) using polynomials. Suppose that one receives $n-1$ of the m packets. Give a method to find a list of size at most p of all possible messages.

(b) Consider an n character message encoded into $m = n+2k$ characters over the field GF(p) using polynomials. Suppose that $k+1$ of the m received packets are corrupted. Give a method to find a list of all possible messages which contain the original message. What is the size of the list for your scheme?

(c) Consider the protocol in (b) where we are working in GF(7). Let the original message have $n = 1$ and $k = 2$, so there are 5 symbols. Now suppose that there are 3 errors, but these three errors all landed on different values. Assume that we received: $0, 0, 1, 2, 3$. How does your list-decoding strategy perform?

Solution:

(a) Since we are trying to encode an n character message using polynomials, we are going to fit our message into a degree $n-1$ polynomial and then encode our message into the length m message $[P(0), P(1), \ldots, P(m-1)]$. Now, we receive $n-1$ of these characters; suppose without loss of generality that the character at position k, $P(k)$ was not received. Now, we know $n-1$ points of the polynomial $P(x)$, but knowing these $n-1$ points gives us no information about $P(k)$ since it is possible for us to construct a degree $n-1$ polynomial that goes through the $n-1$ known points no matter what the value of $P(k)$ is. However, suppose we fix the value of $P(k)$. Then, it turns out that there is exactly one polynomial that goes through the $n-1$ known points and $P(k)$, since a degree $n-1$ polynomial is uniquely determined by n of its points. We use this to deduce that there are at most p different polynomials $P(x)$ that could possibly be our encoding polynomial, one for each possible value of $P(k)$, and thus there are at most p different possible messages that were originally sent. We would generate the p possible messages in the same way:

```
for each value of l in the range 0 to p - 1
    P(x) = interpolate(n - 1 known points, P(k) = l)
    generate the possible message [P(0), P(1), ..., P(m - 1)]
end
```

(b) We can use a similar approach to above; we know that we have $k+1$; if we knew in advance that we were going to have $k+1$ errors we would have sent $n+2k+2$ packets in order to make sure that we could perform error correction using the Berlekamp-Welch method to decode to the correct message. However, we ended up only sending $n+2k$ packets. If we knew the (correct) values of two more packets, then we could use Berlekamp-Welch to decode the message. Since we do not know the values of two more packets, we can do what we did in part (a) and just guess what they are to generate possible messages. Since for the value of each

packets there are p possible values, at most we will generate p^2 possible messages, and the real message will be included.

```
for each value of a in the range 0 to p - 1
    for each value of b in the range 0 to p - 1
        use Berlekamp-Welch with the known values
        and R(n + 2k + 1) = a
        and R(n + 2k + 2) = b
    end
end
```

(c) Given $n = 1$, we have encoded the message into a degree 0 polynomial, so it is just a repetition code. In this case, the list-decoding strategy will pick out the one right answer: 0. Everything else results in 4 errors.