Lagrange interp.

- Secret Sharing
- Erasure codes
- Error correcting codes

$(x_0, y_0), \ldots, (x_d, y_d)$ — deg d poly, GF(p)

k ppl don't give pts → need $d+1+k$ pts

k ppl lie abt pts → $d+1+2k$

run L.I.

poly w/ deg $> d$

(cannot get wrong poly w/ deg $\leq d$)

OR

no sol

need BW

$$Q(x) := P(x)\,E(x)$$

$$E(x_1)\,P(x_1) \equiv E(x_1)\,r_1 \pmod p$$
$$E(x_2)\,P(x_2) \equiv E(x_2)\,r_2$$
$$\vdots$$
$$0 \equiv E(x_i)\,r_i$$
$$0 \equiv 0$$
$$\vdots$$
$$\_\,P(x_{n+2k}) \equiv \_\,r_{n+2k}$$

$E(x_i) = 0$ when equation i is wrong

$E(x_i) \neq 0$ when equation i is good

$n := d+1$

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

(e_k: x-val at error)

CS 70 Discrete Mathematics and Probability Theory
Fall 2020 Dis 05B

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

must P
not corrupted
corrupted

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

n-1
pt pt pt pt

(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.

n+k
n+2k

(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 $Q(x) = P(x)E(x)$?

the x_i corresp. corrupted r_i
No!
n-1+k
k
$Q(x) = P(x)E(x)$ — deg k
(deg n-1)
$n-1+1+k+1 = n+k+1$ don't need! Leading coeff. of $E(x)$ is 1

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

$E(i) = 0$ if $P(i) \neq r_i$
same const
when $P(i) = r_i$, $E(i)P(i) = E(i)r_i$

(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?)

$Q(x)$: $n+k$
$E(x)$: k
$1\ x^k$
Total: $n+2k$ unknowns
$n+2k$ eqns from the points

(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?

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

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.

3 Green Eggs and Hamming

The *Hamming distance* between two length-n bit strings b_1 and b_2 is defined as the minimum number of bits in b_1 you need to flip in order to get b_2. For example, the Hamming distance between 101 and 001 is 1 (since you can just flip the first bit), while the Hamming distance between 111 and 000 is 3 (since you need to flip all three bits).

(a) Sam-I-Am has given you a list of n situations, and wants to know in which of them you would like green eggs and ham. You are planning on sending him your responses encoded in a length n bit string (where a 1 in position i says you would like green eggs and ham in situation i, while a 0 says you would not), but the channel you're sending your answers over is noisy and sometimes corrupts a bit. Sam-I-Am proposes the following solution: you send a length $n+1$ bit string, where the $(n+1)$st bit is the XOR of all the previous n bits (this extra bit is called the parity bit). If you use this strategy, what is the minimum Hamming distance between any two valid bit strings you might send? Why does this allow Sam-I-Am to detect an error? Can he correct the error as well?

(b) If the channel you are sending over becomes more noisy and corrupts two of your bits, can Sam-I-Am still detect the error? Why or why not?

k+1 : so that k flips always lead to invalid string

Assume k for contradiction. If k bits flipped → cannot determine validity

(c) If you know your channel might corrupt up to k bits, what Hamming distance do you need between valid bit strings in order to be sure that Sam-I-Am can detect when there has been a corruption? Prove as well that that your answer is tight—that is, show that if you used a smaller Hamming distance, Sam-I-Am might not be able to detect when there was an error.

(d) Finally, if you want to *correct* up to k corrupted bits, what Hamming distance do you need between valid bit strings? Prove that your condition is sufficient.

2k+1 min Hamming dist

"Codebook" contains valid strings.

Sam finds closest code in the codebook.

b is the correct bit string.

b' is corrupted bit string.

Assume for contra. that dist(c, b') < dist(b, b')

Codebook

1011001 ← 1 away — 1111001

dist(c, b') = ℓ ≤ k

Flip ℓ bits to go from b' to c.

dist(b, b') = m ≤ k

Flip ℓ bits to go from b' to b.

c to b using ℓ + m ≤ 2k flips.

b → b' → c

Contra!