Lecture 10: Error Correcting Codes

WCat Can You 2o Cith A Noisy ChahDel?

Sema-Five?

Suppose I am trying to communicate via semaphore:

Sema-Five?

Suppose I am trying to communicate via semaphore:

What if my recipient misses some letters?

Sema-Five?

Suppose I am trying to communicate via semaphore:

What if my recipient misses some letters?

Less silly: deal with dropped internet packets

Problem Statement

Formally: have message in j parts $h_0; \ldots; h_j$
Channel may drop up to c packets sent
How many packets needed to ensure receipt?

Problem Statement

Formally: have message in j parts $h_0; \ldots; h_j$
Channel may drop up to c packets sent
How many packets needed to ensure receipt?

Naïve idea: repetition coding

- Repeat message "enough times"

Problem Statement

Formally: have message in j parts $h_0; :::; h_j$
Channel may drop up to c packets sent
How many packets needed to ensure receipt?

Naïve idea: repetition coding

- Repeat message "enough times"

How many reps required to $X«9\}9j\,\S 00$ receipt?

Problem Statement

Formally: have message in j parts $h_0; :::; h_j$
Channel may drop up to c packets sent
How many packets needed to ensure receipt?

Naïve idea: repetition coding

- Repeat message "enough times"

How many reps required to $X«9\}9j\,\S00$ receipt?
Could drop first packet every time!
Need $c + 1$ repetitions to be safe

Problem Statement

Formally: have message in j parts $h_0; :::; h_j$
Channel may drop up to c packets sent
How many packets needed to ensure receipt?

Naïve idea: repetition coding

- Repeat message "enough times"

How many reps required to $X«9\}9j\,\S 00$ receipt?
Could drop first packet every time!
Need $c + 1$ repetitions to be safe

For j packet message, send $j(c + 1)$ packets

Problem Statement

Formally: have message in j parts $h_0; :::; h_j$
Channel may drop up to c packets sent
How many packets needed to ensure receipt?

Naïve idea: repetition coding

- Repeat message "enough times"

How many reps required to $X«9\}9j\,\S00$ receipt?
Could drop first packet every time!
Need $c + 1$ repetitions to be safe

For j packet message, send $j(c + 1)$ packets
Can we do better?

A Better Encoding

Ge9_h: Can get away with $j + c$ packets

A Better Encoding

Ge9_h: Can get away with $j + c$ packets

How?

A Better Encoding

Ge9_h: Can get away with $j + c$ packets

How? Using polynomials!

A Better Encoding

Ge9_h: Can get away with $j + c$ packets

How? Using polynomials!

Idea: Take prime x st $x > j + c$, $>$ largest message
Encode message as polynomial in $WS(x)$

A Better Encoding

Ge9_h: Can get away with $j + c$ packets

How? Using polynomials!

Idea: Take prime x st $x > j + c$, $>$ largest message
Encode message as polynomial in $WS(x)$

Interpolate poly $q(^2)$ st $q(\_) = h\_$ for 1 $\_$ j
Send $q(1)$, $q(2)$, ..., $q(j + c)$

Recovery

$\mathbf{Ge9\_h}$: With c erasures, recovery always possible

Recovery

G℮9_h: With c erasures, recovery always possible

p}nn𝕋:

- Suppose receive j points
- Interpolate poly $q^{0}(^{2})$ through them

Recovery

G℮9_h: With c erasures, recovery always possible

p}nn𝕋:

- Suppose receive j points
- Interpolate poly $q'(^2)$ through them
- $\deg(q) = \deg(q') = j \quad 1$
- q and q' agree on j points

Recovery

Ge9_h: With c erasures, recovery always possible

p}nn:

- Suppose receive j points
- Interpolate poly $q'(^2)$ through them
- $\deg(q) = \deg(q') = j \quad 1$
- q and q' agree on j points
- So $q = q'$

Recovery

Ge9_h: With c erasures, recovery always possible

p}nn:

- Suppose receive j points
- Interpolate poly $q'(^2)$ through them
- $\deg(q) = \deg(q') = j \quad 1$
- q and q' agree on j points
- So $q = q'$
- Thus $h_{\_} = q'(\_)$ for $1 \quad \_ \quad j$

Enconding Example

Want to send $h = (4; 0; 5)$, protect for 2 erasures

Enconding Example

Want to send $h = (4; 0; 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

Enconding Example

Want to send $h = (4; 0; 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

$$_{o}(^{2}) = (^{2} \quad 2)(^{2} \quad 3)[(1 \quad 2)(1 \quad 3)]^{o}$$
$$4(^{2^{\copyright}} \quad 5^{2} + 6) \quad 4^{2^{\copyright}} + ^{2} + 3 \pmod 7$$

Enconding Example

Want to send $h = (4; 0; 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

$$_{o}(^{2}) = (^{2} \quad 2)(^{2} \quad 3)[(1 \quad 2)(1 \quad 3)]^{o}$$
$$4(^{2^{\copyright}} \quad 5^{2} + 6) \quad 4^{2^{\copyright}} + ^{2} + 3 \pmod 7$$

Don't need to calculate $_{\copyright}(^{2})$!

Enconding Example

Want to send $h = (4; 0; 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

$$_{o}(^{2}) = (^{2} \quad 2)(^{2} \quad 3)[(1 \quad 2)(1 \quad 3)] \; ^{o}$$
$$4(^{2^{\mathbb{C}}} \quad 5^{2} + 6) \quad 4^{2^{\mathbb{C}}} + {}^{2} + 3 \pmod 7$$

Don't need to calculate $_{\mathbb{C}}(^{2})$!

$$_{\cdot\cdot}(^{2}) = (^{2} \quad 1)(^{2} \quad 2)[(3 \quad 1)(3 \quad 2)] \; ^{o}$$
$$4(^{2^{\mathbb{C}}} \quad 3^{2} + 2) \quad 4^{2^{\mathbb{C}}} + 2^{2} + 1 \pmod 7$$

Enconding Example

Want to send $h = (4; 0; 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

$$_{o}(^{2}) = (^{2} \quad 2)(^{2} \quad 3)[(1 \quad 2)(1 \quad 3)]^{o}$$
$$4(^{2^{\mathbb{C}}} \quad 5^{2} + 6) \quad 4^{2^{\mathbb{C}}} + ^{2} + 3 \pmod 7$$

Don't need to calculate $_{\mathbb{C}}(^{2})$!

$$_{\cdot\cdot}(^{2}) = (^{2} \quad 1)(^{2} \quad 2)[(3 \quad 1)(3 \quad 2)]^{o}$$
$$4(^{2^{\mathbb{C}}} \quad 3^{2} + 2) \quad 4^{2^{\mathbb{C}}} + 2^{2} + 1 \pmod 7$$

$$q(^{2}) = 4 \quad _{o}(^{2}) + 5 \quad _{\cdot\cdot}(^{2}) \quad ^{2^{\mathbb{C}}} + 3 \pmod 7$$

Enconding Example

Want to send $h = (4; 0; 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

$$_{o}(^{2}) = (^{2} \quad 2)(^{2} \quad 3)[(1 \quad 2)(1 \quad 3)]^{o}$$
$$4(^{2^{\mathbb{C}}} \quad 5^{2} + 6) \quad 4^{2^{\mathbb{C}}} + ^{2} + 3 \pmod 7$$

Don't need to calculate $_{\mathbb{C}}(^{2})$!

$$_{\cdot\cdot}(^{2}) = (^{2} \quad 1)(^{2} \quad 2)[(3 \quad 1)(3 \quad 2)]^{o}$$
$$4(^{2^{\mathbb{C}}} \quad 3^{2} + 2) \quad 4^{2^{\mathbb{C}}} + 2^{2} + 1 \pmod 7$$

$$q(^{2}) = 4 \quad _{o}(^{2}) + 5 \quad _{\cdot\cdot}(^{2}) \quad ^{2^{\mathbb{C}}} + 3 \pmod 7$$

Send $(q(1); q(2); q(3); q(4); q(5)) = (4; 0; 5; 5; 0)$

Recovery Example

Sent: $(4; 0; 5; 5; 0)$; Received: $(\quad ; 0; 5; 5; \quad)$

Recovery Example

Sent: $(4; 0; 5; 5; 0)$; Received: $(\quad ; 0; 5; 5; \quad)$

Need to interpolate!

Recovery Example

Sent: $(4; 0; 5; 5; 0)$; Received: $(\quad ; 0; 5; 5; \quad)$
Need to interpolate!

Don't need $\mathbb{C}(^2)$!

Recovery Example

Sent: $(4; 0; 5; 5; 0)$; Received: $(\quad ; 0; 5; 5; \quad)$
Need to interpolate!

Don't need $\quad_{\copyright}(^{2})$!

$$\cdot\cdot(^{2}) = (^{2} \quad 2)(^{2} \quad 4)[(3 \quad 2)(3 \quad 4)]^{\circ}$$
$$6(^{2^{\copyright}} \quad 6^{2} + 8) \quad 6^{2^{\copyright}} + 6^{2} + 6$$

Recovery Example

Sent: $(4; 0; 5; 5; 0)$; Received: $(\quad ; 0; 5; 5; \quad)$
Need to interpolate!

Don't need $\quad_{\mathbb{C}}(^{2})$!

$$\cdot\cdot(^{2}) = (^{2} \quad 2)(^{2} \quad 4)[(3 \quad 2)(3 \quad 4)]^{\circ}$$
$$6(^{2^{\mathbb{C}}} \quad 6^{2} + 8) \quad 6^{2^{\mathbb{C}}} + 6^{2} + 6$$

$$\mathrm{V}(^{2}) = (^{2} \quad 2)(^{2} \quad 3)[(4 \quad 2)(4 \quad 3)]^{\circ}$$
$$4(^{2^{\mathbb{C}}} \quad 5^{2} + 6) \quad 4^{2^{\mathbb{C}}} + {}^{2} + 3$$

Recovery Example

Sent: $(4; 0; 5; 5; 0)$; Received: $(\quad ; 0; 5; 5; \quad)$
Need to interpolate!

Don't need $\quad_{\mathbb{C}}(^{2})$!

$$\cdot\cdot(^{2}) = (^{2} \quad 2)(^{2} \quad 4)[(3 \quad 2)(3 \quad 4)]^{\circ}$$
$$6(^{2^{\mathbb{C}}} \quad 6^{2} + 8) \quad 6^{2^{\mathbb{C}}} + 6^{2} + 6$$

$$_{\mathrm{V}}(^{2}) = (^{2} \quad 2)(^{2} \quad 3)[(4 \quad 2)(4 \quad 3)]^{\circ}$$
$$4(^{2^{\mathbb{C}}} \quad 5^{2} + 6) \quad 4^{2^{\mathbb{C}}} + {}^{2} + 3$$

Interpolate $q^{\emptyset}(^{2}) = 5 \quad \cdot\cdot(^{2}) + 5 \quad _{\mathrm{V}}(^{2}) \quad {}^{2^{\mathbb{C}}} + 3$

Recovery Example

Sent: $(4; 0; 5; 5; 0)$; Received: $(\quad ; 0; 5; 5; \quad)$
Need to interpolate!

Don't need $\quad_{\mathbb{C}}(^{2})$!

$$\cdot\cdot(^{2}) = (^{2} \quad 2)(^{2} \quad 4)[(3 \quad 2)(3 \quad 4)]^{\circ}$$
$$6(^{2^{\mathbb{C}}} \quad 6^{2} + 8) \quad 6^{2^{\mathbb{C}}} + 6^{2} + 6$$

$$_{\mathrm{V}}(^{2}) = (^{2} \quad 2)(^{2} \quad 3)[(4 \quad 2)(4 \quad 3)]^{\circ}$$
$$4(^{2^{\mathbb{C}}} \quad 5^{2} + 6) \quad 4^{2^{\mathbb{C}}} + {}^{2} + 3$$

Interpolate $q^{0}(^{2}) = 5 \quad \cdot\cdot(^{2}) + 5 \quad _{\mathrm{V}}(^{2}) \quad {}^{2^{\mathbb{C}}} + 3$

Evaluate for message: $(q^{0}(1); q^{0}(2); q^{0}(3)) = (4; 0; 5)$

Optimality

Ge9_h: Can't guarantee success w/ $< j + c$ packets

Optimality

G℮9_h: Can't guarantee success w/$< j + c$ packets

p}nn𝕋:

- May send one of two messages:
  - $(h_{\text{o}}; h_{\text{©}}; :::; h_{j\ \text{o}}; h_j)$ or
  - $(h_{\text{o}}; h_{\text{©}}; :::; h_{j\ \text{o}}; h_j^{\emptyset})$

Optimality

Ge9_h: Can't guarantee success w/ $< j + c$ packets

p}nn$\mathbb{T}$:

- May send one of two messages:
  - $(h_{\text{o}}; h_{\text{©}}; :::; h_{j\ \text{o}}; h_j)$ or
  - $(h_{\text{o}}; h_{\text{©}}; :::; h_{j\ \text{o}}; h_j^{\emptyset})$
- Channel drops jth packet and all extras

Optimality

Ge9_h: Can't guarantee success w/ $< j + c$ packets

p}nn$\mathbb{T}$:

- May send one of two messages:
  - $(h_{\text{o}}; h_{\text{©}}; :::; h_{j\ \text{o}}; h_j)$ or
  - $(h_{\text{o}}; h_{\text{©}}; :::; h_{j\ \text{o}}; h_j^{\emptyset})$
- Channel drops jth packet and all extras
- Which message was sent?

Optimality

G℮9_h: Can't guarantee success w/$< j + c$ packets

p}nn𝕋:

- May send one of two messages:
  - $(h_{\text{o}}; h_{\text{©}}; :::; h_{j\ \ \text{o}}; h_j)$ or
  - $(h_{\text{o}}; h_{\text{©}}; :::; h_{j\ \ \text{o}}; h_j^{\emptyset})$
- Channel drops jth packet and all extras
- Which message was sent?
- Impossible to know!

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

Ge9_h: Previous encoding not good enough

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

G℮9_h: Previous encoding not good enough

p}nn𝕋:

- Again, two possible original messages:
  - $(h_0; \ldots; h_{j\ 0}; h_j)$ or
  - $(h_0; \ldots; h_{j\ 0}; h_j^{\emptyset})$

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

G℮9_h: Previous encoding not good enough

p}nn𝕋:

- Again, two possible original messages:
  - $(h_o; :::; h_{j\ \ o}; h_j)$ or
  - $(h_o; :::; h_{j\ \ o}; h_j^{\emptyset})$
- First j rec'd match 1st, but next c match 2nd

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

G℮9_h: Previous encoding not good enough

p}nn𝕋:

- Again, two possible original messages:
  - $(h_0; \ldots; h_{j\ 0}; h_j)$ or
  - $(h_0; \ldots; h_{j\ 0}; h_j^{\emptyset})$
- First j rec'd match 1st, but next c match 2nd
- Which message was sent?

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

G℮9_h: Previous encoding not good enough

p}nn𝕋:

- Again, two possible original messages:
  - $(h_0; \ldots; h_{j\ 0}; h_j)$ or
  - $(h_0; \ldots; h_{j\ 0}; h_j^{\emptyset})$
- First j rec'd match 1st, but next c match 2nd
- Which message was sent?
- Impossible to know!

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

Ge9_h: Previous encoding not good enough

p}nn$\mathbb{T}$:

- Again, two possible original messages:
  - $(h_{\text{o}}; :::; h_{j\ \text{o}}; h_j)$ or
  - $(h_{\text{o}}; :::; h_{j\ \text{o}}; h_j^{\emptyset})$
- First j rec'd match 1st, but next c match 2nd
- Which message was sent?
- Impossible to know!

Note: works for $9j\acute{}$ padding by c packets

NEED MOAR PACKETS

¦ \0n}0 h: For c corruptions, need $j + 2c$ packets

NEED MOAR PACKETS

¦ \0n}0 h: For c corruptions, need $j + 2c$ packets

Suppose only send $2c$ 1 extra packets

NEED MOAR PACKETS

¦ \0n}0 h : For c corruptions, need $j + 2c$ packets

Suppose only send $2c$ 1 extra packets
Consider two possible messages:

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h_j; 0_{\text{o}}; :::; 0_{c\ \text{o}}; 0_c; :::; 0_{\copyright c\ \text{o}})$$

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h'_j; 0'_{\text{o}}; :::; 0'_{c\ \text{o}}; 0'_c; :::; 0'_{\copyright c\ \text{o}})$$

NEED MOAR PACKETS

¦ \0n}0 h : For c corruptions, need $j + 2c$ packets

Suppose only send $2c$ 1 extra packets
Consider two possible messages:

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h_j; 0_{\text{o}}; :::; 0_{c\ \text{o}}; 0_c; :::; 0_{\copyright c\ \text{o}})$$

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h'_j; 0'_{\text{o}}; :::; 0'_{c\ \text{o}}; 0'_c; :::; 0'_{\copyright c\ \text{o}})$$

$$\downarrow$$

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h_j; 0_{\text{o}}; :::; 0_{c\ \text{o}}; 0'_c; :::; 0'_{\copyright c\ \text{o}})$$

NEED MOAR PACKETS

¦ \0n}0 h : For c corruptions, need $j + 2c$ packets

Suppose only send $2c$ 1 extra packets
Consider two possible messages:

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h_j; 0_{\text{o}}; :::; 0_{c\ \text{o}}; \boxed{0_c; :::; 0_{\copyright c\ \text{o}}})$$

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h'_j; 0'_{\text{o}}; :::; 0'_{c\ \text{o}}; 0'_c; :::; 0'_{\copyright c\ \text{o}})$$

$$\downarrow$$

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h_j; 0_{\text{o}}; :::; 0_{c\ \text{o}}; 0'_c; :::; 0'_{\copyright c\ \text{o}})$$

NEED MOAR PACKETS

¦ \0n}0 h : For c corruptions, need $j + 2c$ packets

Suppose only send $2c$ 1 extra packets
Consider two possible messages:

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h_j; 0_{\text{o}}; :::; 0_{c\ \text{o}}; \boxed{0_c; :::; 0_{\copyright c\ \text{o}}})$$

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; \boxed{h'_j; 0'_{\text{o}}; :::; 0'_{c\ \text{o}}}; 0'_c; :::; 0'_{\copyright c\ \text{o}})$$

$$\downarrow$$

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h_j; 0_{\text{o}}; :::; 0_{c\ \text{o}}; 0'_c; :::; 0'_{\copyright c\ \text{o}})$$

NEED MOAR PACKETS

¦ \0n}0 h: For c corruptions, need $j + 2c$ packets

Suppose only send $2c$ 1 extra packets
Consider two possible messages:

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h_j; 0_{\text{o}}; :::; 0_{c\ \text{o}}; \boxed{0_c; :::; 0_{\copyright c\ \text{o}}})$$

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; \boxed{h'_j; 0'_{\text{o}}; :::; 0'_{c\ \text{o}}}; 0'_c; :::; 0'_{\copyright c\ \text{o}})$$

$$\downarrow$$

$$(h_{\text{o}}; h_{\copyright}; :::; h_{j\ \text{o}}; h_j; 0_{\text{o}}; :::; 0_{c\ \text{o}}; 0'_c; :::; 0'_{\copyright c\ \text{o}})$$

Don't know which message originally sent!

Relaaaaax

Take a 4 minute break!

Relaaaaax

Take a 4 minute break!

¦nL9´¸ K_ H« _nj w«0 §_nj:
What's your strangest family tradition?

Corruption Recovery

¦ \0n}0 h: If use previous encoding with $2c$ extra packets, can recover from c corruptions.

Corruption Recovery

¦ \0n}0 h: If use previous encoding with $2c$ extra packets, can recover from c corruptions.

How?

Corruption Recovery

¦ \0n}0 h: If use previous encoding with $2c$ extra packets, can recover from c corruptions.

How? Find deg j 1 poly through $j + c$ points

Corruption Recovery

¦ \0n}0 h: If use previous encoding with $2c$ extra packets, can recover from c corruptions.

How? Find deg j 1 poly through $j + c$ points

Ge9_h: Such a poly exists

Corruption Recovery

¦ \0n}0 h: If use previous encoding with $2c$ extra packets, can recover from c corruptions.

How? Find deg j 1 poly through $j + c$ points

Ge9_h: Such a poly exists

- Original poly through $j + c$ uncorrupted points

Corruption Recovery

¦ \0n}0 h: If use previous encoding with $2c$ extra packets, can recover from c corruptions.

How? Find deg j 1 poly through $j + c$ points

Ge9_h: Such a poly exists

- Original poly through $j + c$ uncorrupted points

Ge9_h: Only one such poly

Corruption Recovery

¦ \0n}0 h: If use previous encoding with $2c$ extra packets, can recover from c corruptions.

How? Find deg j 1 poly through $j + c$ points

Ge9_h: Such a poly exists

- Original poly through $j + c$ uncorrupted points

Ge9_h: Only one such poly

- For any $j + c$ points, at least j uncorrupted
- Those j define the original polynomial

Efficiency?

How long does it take to recover?

Efficiency?

How long does it take to recover?

Naïvely, need to try all possible sets of c corruptions

Efficiency?

How long does it take to recover?

Naïvely, need to try all possible sets of c corruptions $\binom{j+\mathbb{C}c}{c}$ $(\frac{j+\mathbb{C}c}{c})^c$ possibilities — much too slow

Efficiency?

How long does it take to recover?

Naïvely, need to try all possible sets of c corruptions $\binom{j+\mathbb{C}c}{c}$ $(\frac{j+\mathbb{C}c}{c})^c$ possibilities — much too slow

State-of-the-art for over 25 years! (1960 - 1986)

Efficiency?

How long does it take to recover?

Naïvely, need to try all possible sets of c corruptions

$\binom{j+c}{c}$ $\left(\frac{j+c}{c}\right)^c$ possibilities — much too slow

State-of-the-art for over 25 years! (1960 - 1986)

Elwyn Berlekamp

Lloyd Welch

Efficiency?

How long does it take to recover?

Naïvely, need to try all possible sets of c corruptions $\binom{j+c}{c}$ $(\frac{j+c}{c})^c$ possibilities — much too slow

State-of-the-art for over 25 years! (1960 - 1986)

Elwyn Berlekamp

Lloyd Welch

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$$0(^{2}) = (^{2} \quad 0_{\mathrm{o}})(^{2} \quad 0_{\copyright}):::(^{2} \quad 0_{c})$$

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$0(^{2}) = (^{2} \quad 0_{\mathrm{o}})(^{2} \quad 0_{©}):::(^{2} \quad 0_{c})$

If can find this poly, can fix corruptions!

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$0(^{2}) = (^{2} \quad 0_{o})(^{2} \quad 0_{©}):::(^{2} \quad 0_{c})$

If can find this poly, can fix corruptions!

Define (unknown) $x(^{2}) = q(^{2})0(^{2})$ to help solve

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$0(^{2}) = (^{2} \quad 0_{o})(^{2} \quad 0_{©}):::(^{2} \quad 0_{c})$

If can find this poly, can fix corruptions!

Define (unknown) $x(^{2}) = q(^{2})0(^{2})$ to help solve
Claim: $x(\_) = \}\_0(\_)$ for all _

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$$0(^2) = (^2 \quad 0_o)(^2 \quad 0_©):::(^2 \quad 0_c)$$

If can find this poly, can fix corruptions!

Define (unknown) $x(^2) = q(^2)0(^2)$ to help solve
Claim: $x(\_) = \}\_0(\_)$ for all _

- If _ error, both sides zero
- Otherwise $\}\_ = q(\_)$, so true by definition

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$$0(^2) = (^2 \quad 0_o)(^2 \quad 0_©):::(^2 \quad 0_c)$$

If can find this poly, can fix corruptions!

Define (unknown) $x(^2) = q(^2)0(^2)$ to help solve
Claim: $x(\_) = \}\_0(\_)$ for all _

- If _ error, both sides zero
- Otherwise $\}\_ = q(\_)$, so true by definition

Gives $j + 2c$ equations known to be true!
Unknowns are coefficients for $x(^2)$ and $0(^2)$

Berlekamp-Welch: A Closer Look

What does $x(^2)$ look like?

Berlekamp-Welch: A Closer Look

What does $x(^2)$ look like?

$\deg(q) = j \quad 1$, $\deg(0) = c$, so $\deg(x) = j + c \quad 1$

Berlekamp-Welch: A Closer Look

What does $x(^2)$ look like?

$\deg(q) = j \quad 1$, $\deg(0) = c$, so $\deg(x) = j + c \quad 1$

$x(^2) = 9_{j+c \ o}{}^{2j+c \ o} + \cdots + 9_o{}^2 + 9.$

Berlekamp-Welch: A Closer Look

What does $x(^2)$ look like?

$\deg(q) = j \quad 1$, $\deg(0) = c$, so $\deg(x) = j + c \quad 1$

$x(^2) = 9_{j+c \ o}{}^{2j+c \ o} + \cdots + 9_o{}^2 + 9.$

What does $0(^2)$ look like?

Berlekamp-Welch: A Closer Look

What does $x(^2)$ look like?

$\deg(q) = j \quad 1$, $\deg(0) = c$, so $\deg(x) = j + c \quad 1$

$x(^2) = 9_{j+c} \; _{\mathrm{o}}{}^{2j+c} \; ^{\mathrm{o}} + ::: + 9_{\mathrm{o}}{}^2 + 9.$

What does $0(^2)$ look like?

$0(^2) = (^2 \quad 0_{\mathrm{o}})(^2 \quad 0_{©}):::(^2 \quad 0_c)$, so degree c

Berlekamp-Welch: A Closer Look

What does $x(^2)$ look like?

$\deg(q) = j \quad 1$, $\deg(0) = c$, so $\deg(x) = j + c \quad 1$

$x(^2) = 9_{j+c} \; {}_{\mathrm{o}}{}^{2j+c} \; {}^{\mathrm{o}} + \cdots + 9_{\mathrm{o}}{}^2 + 9.$

What does $0(^2)$ look like?

$0(^2) = (^2 \quad 0_{\mathrm{o}})(^2 \quad 0_{\mathrm{©}}) \cdots (^2 \quad 0_c)$, so degree c

$0(^2) = @_c{}^{2c} + \cdots + @_{\mathrm{o}}{}^2 + @.$

Berlekamp-Welch: A Closer Look

What does $x(^2)$ look like?

$\deg(q) = j \quad 1$, $\deg(0) = c$, so $\deg(x) = j + c \quad 1$

$x(^2) = 9_{j+c} \; {}_{\mathrm{o}}{}^{2j+c} \; {}^{\mathrm{o}} + ::: + 9_{\mathrm{o}}{}^2 + 9.$

What does $0(^2)$ look like?

$0(^2) = (^2 \quad 0_{\mathrm{o}})(^2 \quad 0_{\copyright}):::(^2 \quad 0_c)$, so degree c

$0(^2) = @_c{}^{2c} + ::: + @_{\mathrm{o}}{}^2 + @.$

But wait! $@_c = 1$ for any $0_{\mathrm{o}}; :::; 0_c$!

So $0(^2) = {}^{2c} + @_c \; {}_{\mathrm{o}}{}^{2c} \; {}^{\mathrm{o}} + ::: + @_{\mathrm{o}}{}^2 + @.$

Berlekamp-Welch: A Closer Look

What does $x(^2)$ look like?

$\deg(q) = j \quad 1$, $\deg(0) = c$, so $\deg(x) = j + c \quad 1$

$x(^2) = 9_{j+c} \; {}_{\mathrm{o}}{}^{2j+c} \; {}^{\mathrm{o}} + \cdots + 9_{\mathrm{o}}{}^2 + 9.$

What does $0(^2)$ look like?

$0(^2) = (^2 \quad 0_{\mathrm{o}})(^2 \quad 0_{\copyright})\cdots(^2 \quad 0_c)$, so degree c

$0(^2) = @_c{}^{2c} + \cdots + @_{\mathrm{o}}{}^2 + @.$

But wait! $@_c = 1$ for any $0_{\mathrm{o}}; \ldots; 0_c$!

So $0(^2) = {}^{2c} + @_c \; {}_{\mathrm{o}}{}^{2c} \; {}^{\mathrm{o}} + \cdots + @_{\mathrm{o}}{}^2 + @.$

Have $j + c$ unknowns from x, c from 0

Matches $j + 2c$ linear eqns of the form $x(\_) = \}\_0(\_)$

Berlekamp-Welch: A Closer Look

What does $x(^2)$ look like?
$\deg(q) = j \quad 1$, $\deg(0) = c$, so $\deg(x) = j + c \quad 1$
$x(^2) = 9_{j+c}{}_{\text{o}}{}^{2j+c}{}^{\text{o}} + :::: + 9_{\text{o}}{}^2 + 9.$

What does $0(^2)$ look like?
$0(^2) = (^2 \quad 0_{\text{o}})(^2 \quad 0_{\copyright}):::(^2 \quad 0_c)$, so degree c
$0(^2) = @_c{}^{2c} + :::: + @_{\text{o}}{}^2 + @.$

But wait! $@_c = 1$ for any $0_{\text{o}}; ::::; 0_c$!
So $0(^2) = {}^{2c} + @_c{}_{\text{o}}{}^{2c}{}^{\text{o}} + :::: + @_{\text{o}}{}^2 + @.$

Have $j + c$ unknowns from x, c from 0
Matches $j + 2c$ linear eqns of the form $x(\_) = \}\_0(\_)$

Linear Algebra: can find x, 0, so have $q(^2) = \frac{x(^2)}{0(^2)}$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1; 3)$, $(2; 1)$, $(3; 4)$, $(4; 0)$ mod 7

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1;3)$, $(2;1)$, $(3;4)$, $(4;0)$ mod 7

$x(^2) = 9_{©}{}^{2^{©}} + 9_{o}{}^{2} + 9.$, $0(^2) = {}^{2} + @.$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1;3)$, $(2;1)$, $(3;4)$, $(4;0)$ mod 7

$x(²) = 9_{©}{}^{2^{©}} + 9_{o}{}^{²} + 9.$, $0(²) = {}^{²} + @.$

Eq 1: $x(1) = \}_{o}0(1)$, so $9_{©} + 9_{o} + 9. = 3(1 + @.)$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1;3)$, $(2;1)$, $(3;4)$, $(4;0)$ mod 7

$x(²) = 9_{©}{}^{2©} + 9_{o}{}^{2} + 9.$, $0(²) = {}^{2} + @.$

Eq 1: $x(1) = \}_{o}0(1)$, so $9_{©} + 9_{o} + 9. = 3(1 + @.)$

Eq 2: $x(2) = \}_{©}0(2)$, so $49_{©} + 29_{o} + 9. = 1(2 + @.)$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1;3)$, $(2;1)$, $(3;4)$, $(4;0)$ mod 7

$x(²) = 9_{©}{}^{2©} + 9_{o}{}^{2} + 9.$, $0(²) = {}^{2} + @.$

Eq 1: $x(1) = \}_{o}0(1)$, so $9_{©} + 9_{o} + 9. = 3(1 + @.)$

Eq 2: $x(2) = \}_{©}0(2)$, so $49_{©} + 29_{o} + 9. = 1(2 + @.)$

Eq 3: $x(3) = \}_{\cdot\cdot}0(3)$, so $99_{©} + 39_{o} + 9. = 4(3 + @.)$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1;3)$, $(2;1)$, $(3;4)$, $(4;0)$ mod 7

$x(\text{²}) = 9_{\text{©}}\text{²}^{\text{©}} + 9_{\text{o}}\text{²} + 9.$, $0(\text{²}) = \text{²} + @.$

Eq 1: $x(1) = \}_{\text{o}}0(1)$, so $9_{\text{©}} + 9_{\text{o}} + 9. = 3(1 + @.)$

Eq 2: $x(2) = \}_{\text{©}}0(2)$, so $49_{\text{©}} + 29_{\text{o}} + 9. = 1(2 + @.)$

Eq 3: $x(3) = \}_{\cdot\cdot}0(3)$, so $99_{\text{©}} + 39_{\text{o}} + 9. = 4(3 + @.)$

Eq 4: $x(4) = \}_{\text{V}}0(4)$, so $169_{\text{©}} + 49_{\text{o}} + 9. = 0(4 + @.)$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1;3)$, $(2;1)$, $(3;4)$, $(4;0)$ mod 7

$x(^2) = 9_{\copyright}{}^{2^{\copyright}} + 9_o{}^2 + 9.$, $0(^2) = {}^2 + @.$

Eq 1: $x(1) = \}_o 0(1)$, so $9_{\copyright} + 9_o + 9. = 3(1 + @.)$

Eq 2: $x(2) = \}_{\copyright} 0(2)$, so $49_{\copyright} + 29_o + 9. = 1(2 + @.)$

Eq 3: $x(3) = \}_{\cdot\cdot} 0(3)$, so $99_{\copyright} + 39_o + 9. = 4(3 + @.)$

Eq 4: $x(4) = \}_V 0(4)$, so $169_{\copyright} + 49_o + 9. = 0(4 + @.)$

Note: all eqns modulo 7, so can shrink some nums

(Berlekamp-Welch: Example): Continued

Simplify equations mod 7, move all variables to left:

$9_{©} + 9_{o} + 9. \quad 3@. = 3$

$49_{©} + 29_{o} + 9. \quad @. = 2$

$29_{©} + 39_{o} + 9. \quad 4@. = 5$

$29_{©} + 49_{o} + 9. = 0$

(Berlekamp-Welch: Example): Continued

Simplify equations mod 7, move all variables to left:

$9_{©} + 9_{o} + 9. \quad 3@. = 3$

$49_{©} + 29_{o} + 9. \quad @. = 2$

$29_{©} + 39_{o} + 9. \quad 4@. = 5$

$29_{©} + 49_{o} + 9. = 0$

Can use Gaussian Elimination (mod 7) to solve

(Berlekamp-Welch: Example): Continued

Simplify equations mod 7, move all variables to left:

$9_{©} + 9_{o} + 9. \quad 3@. = 3$

$49_{©} + 29_{o} + 9. \quad @. = 2$

$29_{©} + 39_{o} + 9. \quad 4@. = 5$

$29_{©} + 49_{o} + 9. = 0$

Can use Gaussian Elimination (mod 7) to solve

Here, $9_{©} = 3$, $9_{o} = 6$, $9. = 5$, $@. = 6$

So $x(^{2}) = 3^{2^{©}} + 6^{2} + 5$, $0(^{2}) = {}^{2} + 6$

(Berlekamp-Welch: Example): Continued

Simplify equations mod 7, move all variables to left:

$9_{©} + 9_{o} + 9. \quad 3@. = 3$

$49_{©} + 29_{o} + 9. \quad @. = 2$

$29_{©} + 39_{o} + 9. \quad 4@. = 5$

$29_{©} + 49_{o} + 9. = 0$

Can use Gaussian Elimination (mod 7) to solve

Here, $9_{©} = 3$, $9_{o} = 6$, $9. = 5$, $@. = 6$

So $x(^{2}) = 3^{2^{©}} + 6^{2} + 5$, $0(^{2}) = {}^{2} + 6$

Do poly long division mod 7 to get $q(^{2}) = 3^{2} + 2$

Original messages: $q(1) = 5$, $q(2) = 1$

Fin

Next time: countability!