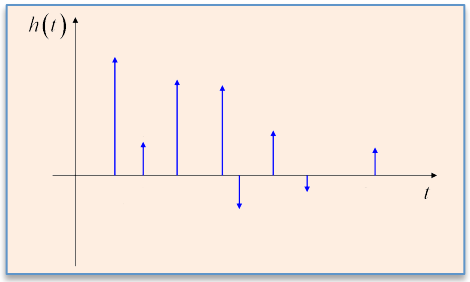
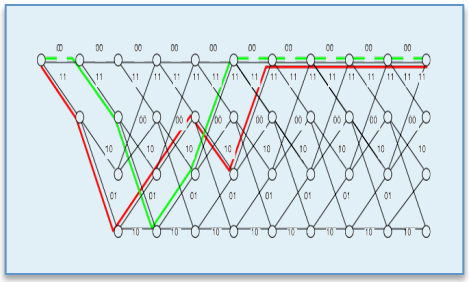
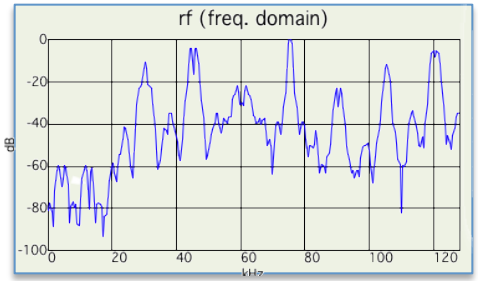
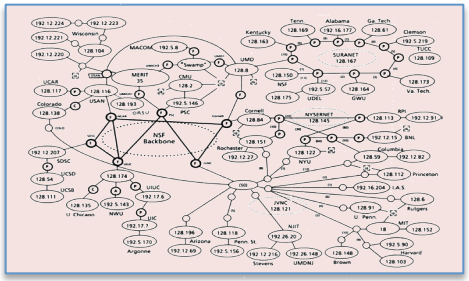








INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

6.02 Fall 2012
Lecture #5

- Hamming codes
- Error correction for linear block codes
  - Syndrome decoding
- Burst errors and interleaving

Hamming Codes

- Hamming codes correct single errors with the minimum number of parity bits:

$$2^{n-k} = n + 1$$

so n+1 must be a power of 2.

- (7,4,3)
- (15,11,3)
- ($n=2^m-1$, $k=n-m$, 3)

- Such efficiency is not the only, or even most important, criterion in picking a good code. The ability of a code's k/n to approach channel capacity, and various other factors, are important.

Constraints for more than single-bit errors

Parity constraint inequality for locating **single-bit** errors

$$1 + n \leq 2^{n-k}$$

Write-out the inequality for **t** bit errors

Elementary Combinatorics

- Given n objects, in how many ways can we choose m of them?

If the ordering of the m selected objects matters, then

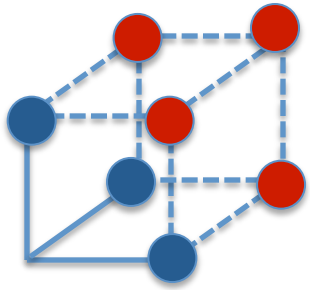
n(n-1)(n-2) … (n-m+1) = n!/(n-m)!

If the ordering of the m selected objects doesn't matter, then the above expression is too large by a factor m!, so

"n choose m" = $$\binom{n}{m} = \frac{n!}{(n-m)!m!}$$

Logic Behind Hamming Code Construction – (7,4,3) example

- Idea: Use parity bits to cover each axis of the binary vector space that represents "parity coverage" (this is NOT the space of codewords!)
  - That way, all message bits will be monitored by a **unique** combination of parity bits



Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Binary index	001	010	011	100	101	110	111
(7,4) code	**P1**	**P2**	**D1**	**P3**	**D2**	**D3**	**D4**

$P_1 = D_1+D_2+D_4$
$P_2 = D_1+D_3+D_4$
$P_3 = D_2+D_3+D_4$

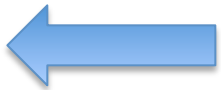
P_1 with binary index 00**1** covers

D_1 with binary index 01**1**
D_2 with binary index 10**1**
D_4 with binary index 11**1**

Syndrome Decoding

- After receiving the possibly corrupted message (use ' to indicate possibly erroneous symbol), compute a **syndrome** bit (S_i) for each parity bit

$$S_1 = D'_1 + D'_2 + D'_4 + P'_1$$
$$S_2 = D'_1 + D'_3 + D'_4 + P'_2$$
$$S_3 = D'_2 + D'_3 + D'_4 + P'_3$$



$$0 = D_1+D_2+D_4+P_1$$
$$0 = D_1+D_3+D_4+P_2$$
$$0 = D_2+D_3+D_4+P_3$$

- If all the S_i are zero: no errors
- Otherwise use the particular combination of the S_i to figure out correction

Index	1	2	3	4	5	6	7
Binary index	001	010	011	100	101	110	111
(7,4) code	P1	P2	D1	P3	D2	D3	D4

$S_3S_2S_1$	Corrective Action
000	no errors
001	p_1 has an error, flip to correct
010	p_2 has an error, flip to correct
011	d_1 has an error, flip to correct
100	p_3 has an error, flip to correct
101	d_2 has an error, flip to correct
110	d_3 has an error, flip to correct
111	d_4 has an error, flip to correct

… versus Simple-minded Decoding

- Compare received n-bit word R = C + E (where E is channel-noise vector) against each of 2^k valid codewords to see which one is 0 or 1 away in Hamming distance.

- Doesn't exploit the nice structure of the code!

What happens if there are TWO errors?

- $S_1 = D'_1 + D'_2 + D'_4 + P'_1$
- $S_2 = D'_1 + D'_3 + D'_4 + P'_2$
- $S_3 = D'_2 + D'_3 + D'_4 + P'_3$

Suppose D_2 **AND** D_3 are in error?

Matrix Notation for Linear Block Codes

Task: given k-bit message, compute n-bit codeword. We can use standard matrix arithmetic (modulo 2) to do the job. For example, here's how we would describe the (9,4,4) rectangular code that includes an overall parity bit.

$$D \cdot G = C$$

← Back to notation in notes!

$$\begin{bmatrix} D_1 & D_2 & D_3 & D_4 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} D_1 & D_2 & D_3 & D_4 & P_1 & P_2 & P_3 & P_4 & P_5 \end{bmatrix}$$

1×k message vector

k×n generator matrix

1×n code word vector in the **row space** of G

The generator matrix, $G_{kxn} = \left[\begin{array}{c|c} I_{k\times k} & A_{k\times(n-k)} \end{array}\right]$

Write out G for Hamming (7,4,3)

$$P_1 = D_1+D_2+D_4$$
$$P_2 = D_1+D_3+D_4$$
$$P_3 = D_2+D_3+D_4$$

How does G change if we now add an overall parity bit?

$$P_o = D_1+ D_2+ D_3 + D_4+ P_1+ P_2+P_3$$

$$= D_1+ D_2+ D_3 \quad \text{(why?)}$$

A closer look at the Parity Matrix A

Parity equation $$P_j = \sum_{i=1}^{k} D_i a_{ij}$$

Parity relation $$P_j + \sum_{i=1}^{k} D_i a_{ij} = 0$$

$$A = [a_{ij}]$$

So entry a_{ij} in i-th row, j-th column of A specifies whether data bit D_i is used in constructing parity bit P_j

Questions: Can two columns of A be the same? Should two columns of A be the same? How about rows?

Parity Check Matrix H

Can restate the codeword generation process as a parity check or **nullspace** check

$$H_{(n-k)xn} \cdot C^{T}_{1xn} = 0$$

The parity check matrix,

$$H = A^{T} | I_{(n-k)\times(n-k)}$$

For (9,4,4) example

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{bmatrix} = 0_{5x1}$$

(n-k) x n parity check matrix

n×1 code word vector (transpose)

Extracting min HD (=d) from H

- Claim: The minimum HD (=d) of the linear code is the minimum number of columns of H that are linearly dependent, i.e., that can be combined to give the zero vector

- Proof: d = minimum-weight nonzero codeword

- One consequence: If A has two identical rows, then A^T has two identical columns, which means d is no greater than 2, so error correction is not possible.

- Confirm for the the Hamming (8,4,4) case, i.e., (7,4,3) with an overall parity bit.

Syndrome Decoding – Matrix Form

Task: given n-bit code word, compute (n-k) syndrome bits. Again we can use matrix multiplication to do the job:

received word $R = C + E$

Compute syndromes on receive word $H \cdot R^T = S$

(n-k) x 1 syndrome vector

To figure out the relationship of syndromes to errors:

$$H \cdot (C + E)^T = S \qquad \text{use} \qquad H \cdot C^T = 0$$

$$H \cdot E^T = S$$

figure-out error type from Syndrome

Knowing the error patterns we want to correct for, we can compute k syndrome vectors offline (or n, if you want to correct errors in the parity bits, but this is not needed) and then do a lookup after the syndrome is calculated from a received word to find the error type that occurred

Syndrome Decoding - Steps

Step 1: For a given code and error patterns E_i, precompute syndromes and store them:

$$H \cdot E_i = S_i$$

For error patterns with just a single 1 in the i-th position, this is trivial --- just the i-th column of H.

Step 2: For each received word, compute the Syndrome

$$H \cdot R = S$$

Step 3: Find *l* such that $S_l = S$ and apply correction for error E_l

$$C = R + E_l$$

Syndrome Decoding - (9,4,4) example

Codeword generation:

$$\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}\cdot\begin{bmatrix}1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1\\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1\\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1\\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

Received word in error:

$$\begin{bmatrix}1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0\end{bmatrix}=\begin{bmatrix}1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0\end{bmatrix}+\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

Syndrome computation for received word

$$\begin{bmatrix}1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0\\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}1\\0\\1\\1\\0\\0\\0\\0\\0\end{bmatrix}=\begin{bmatrix}1\\0\\0\\1\\1\end{bmatrix}$$

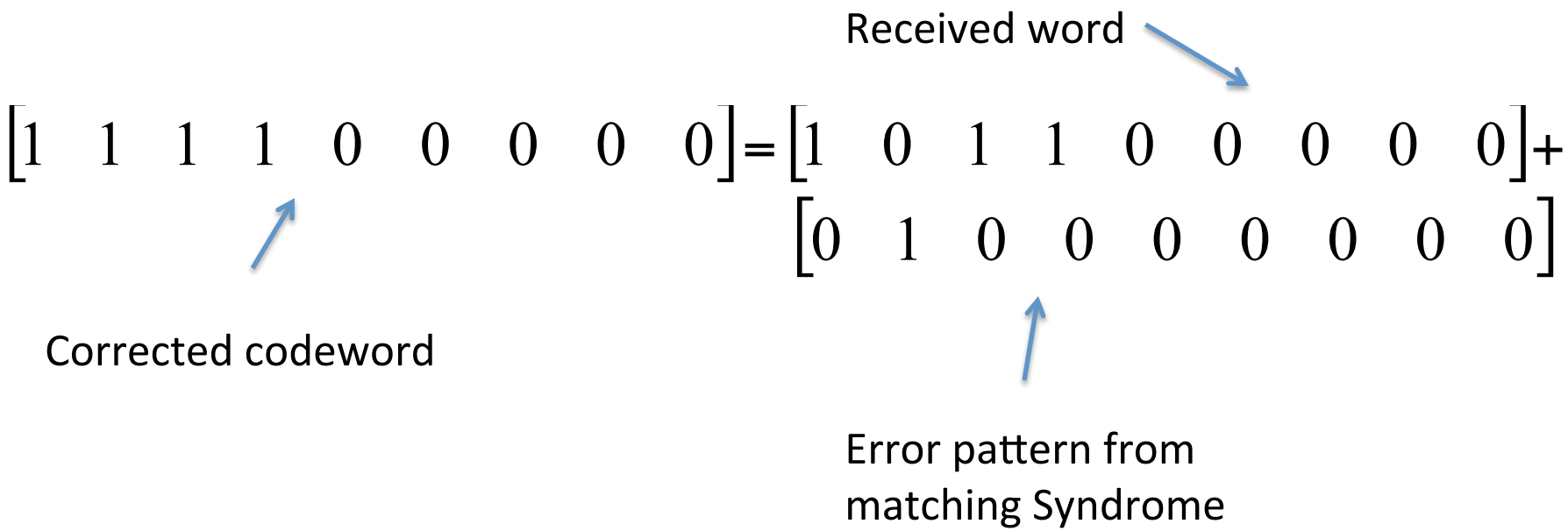
Precomputed Syndrome for a given error pattern

$$\begin{bmatrix}1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0\\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}=\begin{bmatrix}1\\0\\0\\1\\1\end{bmatrix}$$

Syndrome Decoding – (9,4,4) example

Correction:

Since received word syndrome $[1\ 0\ 0\ 1\ 1]^T$ matches the
syndrome of the error [0 1 0 0 0 0 0 0 0],
apply this error to the received word to recover the original codeword



Linear Block Codes: Wrap-Up

- (n,k,d) codes have rate k/n and can correct up to t errors, provide $d \geq 2t + 1$
- Code bits are linear combinations of message bits: sum of any two code words is a code word
  - Message + 1 parity bit: (n+1,n,2) code
    - Good code rate, but only 1-bit error detection
  - Replicating each bit c times is a (c,1,c) code
    - Simple way to get great error correction; poor code rate
  - Hamming single-error correcting codes are (n, n-m, 3) where $n = 2^m - 1$ for $m > 1$
    - Adding an overall parity bit makes the code (n+1,n-m,4)
  - Rectangular parity codes are (rc+r+c, rc, 3) codes
    - Rate not as good as Hamming codes
- Syndrome decoding: general efficient approach for decoding linear block codes

Burst Errors

- Correcting single-bit errors is good
- Similar ideas could be used to correct independent multi-bit errors
- But in many situations errors come in bursts: correlated multi-bit errors (e.g., fading or burst of interference on wireless channel, damage to storage media etc.). How does single-bit error correction help with that?

Independent multi-bit errors

e.g., m errors in n bits

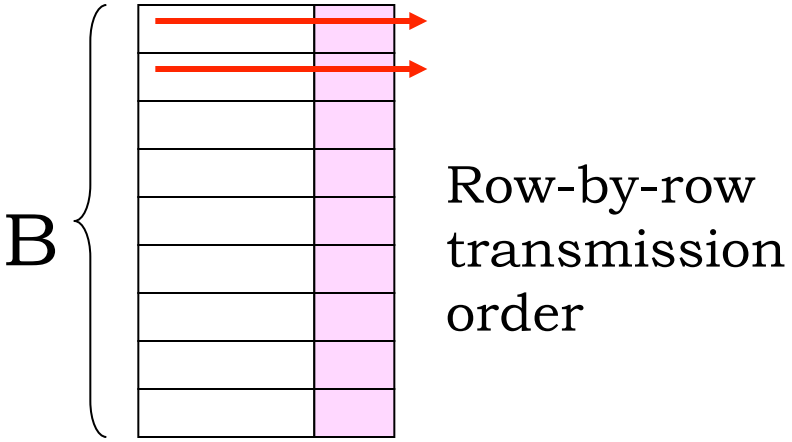
$$\binom{n}{m} p^m (1-p)^{n-m}$$

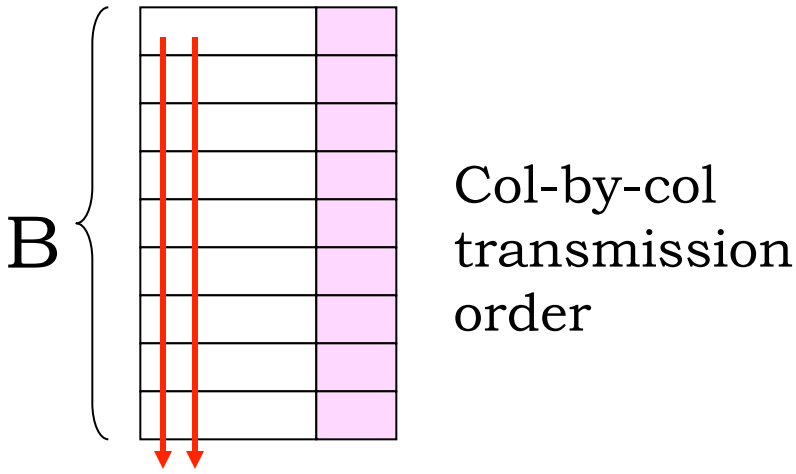
$$\binom{n}{m} = \frac{n!}{(n-m)!(m)!}$$

Stirling's approximation: $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

Coping with Burst Errors by Interleaving

Can we think of a way to turn a B-bit error burst into B single-bit errors?





Problem: Bits from a particular codeword are transmitted sequentially, so a B-bit burst produces multi-bit errors.

Solution: interleave bits from B different codewords. Now a B-bit burst produces 1-bit errors in B different codewords.