15-853:Algorithms in the Real World

Error Correcting Codes I
- Overview
- Hamming Codes
- Linear Codes

Error Correcting Codes II (Reed-Solomon Codes)

Error Correcting Codes III (LDPC/Expander Codes)

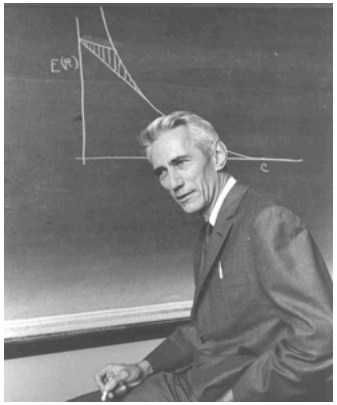
A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist[1] and Hartley[2] on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the



message. In a multiplex PCM system the different speech functions must be sampled, compressed, quantized and encoded, and finally interleaved

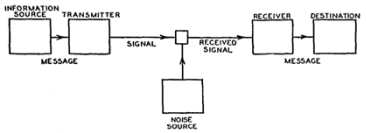


Fig. 1—Schematic diagram of a general communication system.

properly to construct the signal. Vocoder systems, television, and frequency modulation are other examples of complex operations applied to the message to obtain the signal.

3. The *channel* is merely the medium used to transmit the signal from transmitter to receiver. It may be a pair of wires, a coaxial cable, a band of radio frequencies, a beam of light, etc.

4. The *receiver* ordinarily performs the inverse operation of that done by the transmitter, reconstructing the message from the signal.

5. The *destination* is the person (or thing) for whom the message is intended.

In Appendix II, the following result is established:

Theorem 2: The only H satisfying the three above assumptions is of the form:

$$H = -K \sum_{i=1}^{n} p_i \log p_i$$

where K is a positive constant.

This theorem, and the assumptions required for its proof, are in no way necessary for the present theory. It is given chiefly to lend a certain plausibility to some of our later definitions. The real justification of these definitions, however, will reside in their implications.

Quantities of the form $H = -\Sigma p_i \log p_i$ (the constant K metely amounts to a choice of a unit of measure) play a central role in information theory as measures of information, choice and uncertainty. The form of H will be

To give a visual idea of how this series of processes approaches a language, typical sequences in the approximations to English have been constructed and are given below. In all cases we have assumed a 27-symbol "alphabet," the 26 letters and a space.

1. Zero-order approximation (symbols independent and equi-probable).
   XFOML RXKHRJFFJUJ ZLPWCFWKCYJ
   FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD
2. First-order approximation (symbols independent but with frequencies of English text).
   OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI
   ALHENHTTPA OOBTTVA NAH BRL
3. Second-order approximation (digram structure as in English).
   ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY
   ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO
   TIZIN ANDY TOBE SEACE CTISBE
4. Third-order approximation (trigram structure as in English).
   IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID
   PONDENOME OF DEMONSTURES OF THE REPTAGIN IS
   REGOACTIONA OF CRE
5. First-Order Word Approximation. Rather than continue with tetragram, · · ·, n-gram structure it is easier and better to jump at this point to word units. Here words are chosen independently but with their appropriate frequencies.
   REPRESENTING AND SPEEDILY IS AN GOOD APT OR
   COME CAN DIFFERENT NATURAL HERE HE THE A IN
   CAME THE TO OF TO EXPERT GRAY COME TO FUR-
   NISHES THE LINE MESSAGE HAD BE THESE.
6. Second-Order Word Approximation. The word transition probabilities are correct but no further structure is included.
   THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH
   WRITER THAT THE CHARACTER OF THIS POINT IS
   THEREFORE ANOTHER METHOD FOR THE LETTERS
   THAT THE TIME OF WHO EVER TOLD THE PROBLEM
   FOR AN UNEXPECTED

The resemblance to ordinary English text increases quite noticeably at each of the above steps. Note that these samples have reasonably good structure out to about twice the range that is taken into account in their

The Bell System Technical Journal

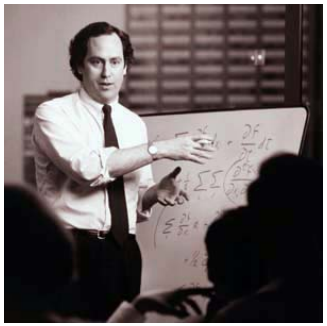
Vol. XXIX *April, 1950* *No. 2*

Error Detecting and Error Correcting Codes

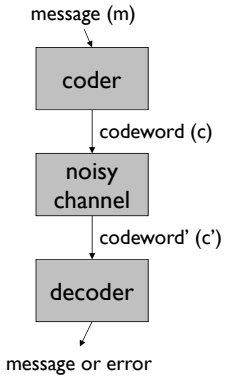
By R. W. HAMMING

1. Introduction

THE author was led to the study given in this paper from a consideration of large scale computing machines in which a large number of operations must be performed without a single error in the end result. This problem of "doing things right" on a large scale is not essentially new; in a telephone central office, for example, a very large number of operations are performed while the errors leading to wrong numbers are kept well under control, though they have not been completely eliminated. This has been achieved, in part, through the use of self-checking circuits. The occasional failure that escapes routine checking is still detected by the customer and will, if it persists, result in customer complaint, while if it is transient it will produce only occasional wrong numbers. At the same time the rest of the central office functions satisfactorily. In a digital computer, on the other hand, a single failure usually means the complete failure, in the sense that if it is detected no more computing can be done until the failure is located



General Model



Errors introduced by the noisy channel:

- changed fields in the codeword (e.g. a flipped bit)
- missing fields in the codeword (e.g. a lost byte). Called **erasures**

How the decoder deals with errors.

- **error detection** vs.
- **error correction**

Applications

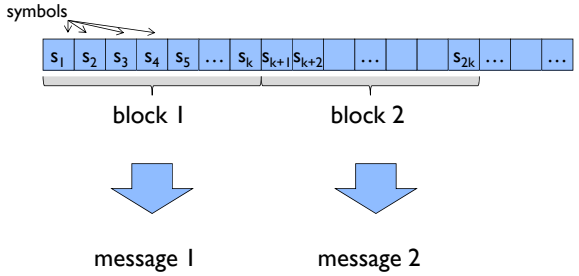
- **Storage**: CDs, DVDs, cloud storage, NAND flash…
- **Wireless**: Cell phones, wireless links
- **Satellite and Space**: TV, Mars rover, …
- **Digital Television**: DVD, MPEG2 layover
- **High Speed Modems**: ADSL, DSL, ..

Reed-Solomon codes are by far the most used in practice, including pretty much all the examples mentioned above.

LDPC codes used for 4G communication.

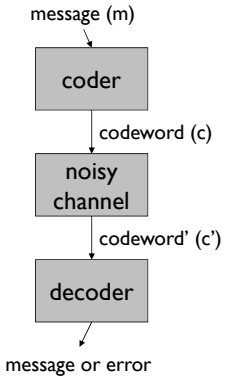
Algorithms for decoding are quite sophisticated.

Block Codes



Other kind: convolutional codes (we won't cover it)…

Block Codes



Each message and codeword is of fixed size

Σ = codeword alphabet

$\mathbf{k} = |m|$ $\mathbf{n} = |c|$ $\mathbf{q} = |\Sigma|$

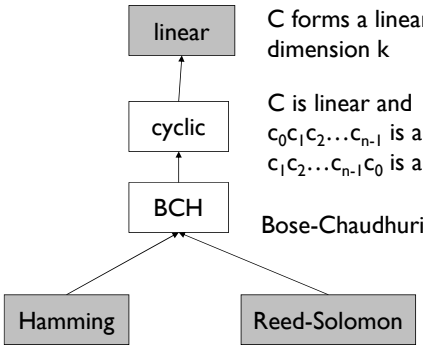
$\mathbf{C} \subseteq \Sigma^n$ (codewords)

$\Delta(\mathbf{x},\mathbf{y})$ = number of positions s.t. $x_i \neq y_i$

$\mathbf{d} = \min\{\Delta(x,y) : x,y \in C, x \neq y\}$

Code described as: $\mathbf{(n,k,d)_q}$

Hierarchy of Codes



C forms a linear subspace of $\sum^n$ of dimension k

C is linear and
$c_0c_1c_2\ldots c_{n-1}$ is a codeword implies
$c_1c_2\ldots c_{n-1}c_0$ is a codeword

Bose-Chaudhuri-Hochquenghem

These are all block codes.

Binary Codes

Today we will mostly be considering $\sum$ **= {0,1}** and will sometimes use (n,k,d) as shorthand for $(n,k,d)_2$

In binary $\Delta(x,y) = |\{ i : x_i \neq y_i \}|$
is often called the **Hamming distance**

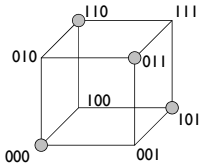
Example of $(6,3,3)_2$ systematic code

message	codeword
000	**000**000
001	**001**011
010	**010**101
011	**011**110
100	**100**110
101	**101**101
110	**110**011
111	**111**000

Definition: A **Systematic code** is one in which the message appears in the codeword

Hypercube Interpretation

Consider codewords as vertices on a hypercube.



codeword

d = 2 = min distance
n = 3 = dimensionality
2^n = 8 = number of nodes

The distance between nodes on the hypercube is the Hamming distance Δ. The minimum distance is d.

001 is equidistant from 000, 011 and 101.

For s-bit error detection $d \geq s + 1$

For s-bit error correction $d \geq 2s + 1$

Error Detection with Parity Bit

A $(k+1,k,2)_2$ systematic code

Encoding:

$m_1m_2\ldots m_k \Rightarrow m_1m_2\ldots m_kp_{k+1}$

where $p_{k+1} = m_1 \oplus m_2 \oplus \ldots \oplus m_k$

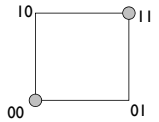
d = 2 since the parity is always even (it takes two bit changes to go from one codeword to another).

Detects one-bit error since this gives odd parity

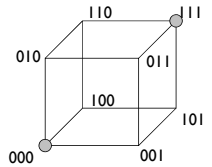
Cannot be used to correct 1-bit error since any odd-parity word is equal distance Δ to k+1 valid codewords.

Error Correcting One Bit Messages

How many bits do we need to correct a one bit error on a one bit message?



2 bits
0 -> 00, 1-> 11
(n=2,k=1,d=2)



3 bits
0 -> 000, 1-> 111
(n=3,k=1,d=3)

In general need $d \geq 3$ to correct one error. Why?

Desiderata

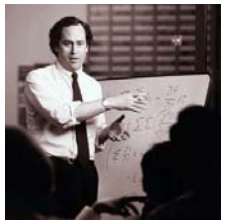
We look for codes with the following properties:

1. Good rate: k/n should be high (low overhead)
2. Good distance: d should be large (good error correction)
3. Small block size k
4. Fast encoding and decoding
5. Others: want to handle bursty/random errors, local decodability, network effects, …

Error Correcting Multibit Messages

We will first discuss **Hamming Codes**

Named after Richard Hamming (1915-1998), a pioneer in error-correcting codes and computing in general.



Aside: has a lecture called *You and Your Research* that is an interesting read (or you can listen to it on YouTube).

Error Correcting Multibit Messages

We will first discuss **Hamming Codes**
Detect 2-bit errors, or correct 1-bit errors.

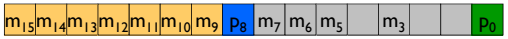
Codes are of form: $(2^r-1, 2^r-1-r, 3)$ for any $r > 1$
e.g. (3,1,3), (7,4,3), (15,11,3), (31, 26, 3), …
which correspond to 2, 3, 4, 5, … "parity bits" (i.e. n-k)

The high-level idea is to "localize" the error.

Any specific ideas?

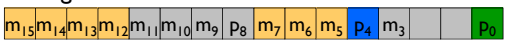
Hamming Codes: Encoding

Localizing error to top or bottom half 1xxx or 0xxx



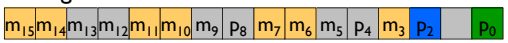
$p_8 = m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_{11} \oplus m_{10} \oplus m_9$

Localizing error to x1xx or x0xx



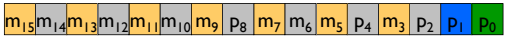
$p_4 = m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_7 \oplus m_6 \oplus m_5$

Localizing error to xx1x or xx0x



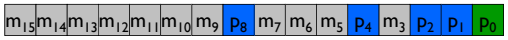
$p_2 = m_{15} \oplus m_{14} \oplus m_{11} \oplus m_{10} \oplus m_7 \oplus m_6 \oplus m_3$

Localizing error to xxx1 or xxx0



$p_1 = m_{15} \oplus m_{13} \oplus m_{11} \oplus m_9 \oplus m_7 \oplus m_5 \oplus m_3$

Hamming Codes: Decoding



We don't need p_0, so we have a (15,11,?) code.
After transmission, we generate

$b_8 = p_8 \oplus m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_{11} \oplus m_{10} \oplus m_9$
$b_4 = p_4 \oplus m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_7 \oplus m_6 \oplus m_5$
$b_2 = p_2 \oplus m_{15} \oplus m_{14} \oplus m_{11} \oplus m_{10} \oplus m_7 \oplus m_6 \oplus m_3$
$b_1 = p_1 \oplus m_{15} \oplus m_{13} \oplus m_{11} \oplus m_9 \oplus m_7 \oplus m_5 \oplus m_3$

With no errors, these will all be zero
With one error $b_8b_4b_2b_1$ gives us the error location.
e.g. **0100** would tell us that $\mathbf{p_4}$ is wrong, and
1100 would tell us that $\mathbf{m_{12}}$ is wrong

Hamming Codes

Can be generalized to any power of 2

- $n = 2^r - 1$ (15 in the example)
- (n-k) = r (4 in the example)
- $d \geq 3$ (since we can correct one error)
- Can correct one error,
  but can't tell difference between one and two!
- Gives $(2^r-1, 2^r-1-r, 3)$ code

Extended Hamming code

- Add back the parity bit at the end
- Gives $(2^r, 2^r-1-r, 4)$ code
- Can still correct one error, but now can detect 2

Lower bound on parity bits

How many nodes in hypercube do we need so that d = 3?

Each of 2^k codewords eliminates n neighbors plus itself, i.e. n+1

$$\begin{aligned} 2^n &\geq (n+1)2^k \\ n &\geq k + \log_2(n+1) \\ n &\geq k + \lceil \log_2(n+1) \rceil \end{aligned}$$

In above Hamming code, $15 \geq 11 + \lceil \log_2(15+1) \rceil = 15$.

Hamming Codes are called **perfect codes** since they match the lower bound exactly.

Lower bound on parity bits

What about fixing 2 errors (i.e. d=5)?

Each of the 2^k codewords eliminates itself, its neighbors and its neighbors' neighbors, giving: $1+\binom{n}{1}+\binom{n}{2}$

$$\begin{aligned} 2^n &\geq (1+n+n(n-1)/2)2^k \\ n &\geq k + \log_2(1+n+n(n-1)/2) \\ &\geq k + 2\log_2 n - 1 \end{aligned}$$

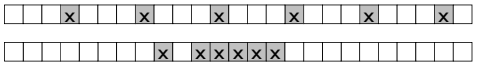
Generally to correct s errors:

$$n \geq k + \log_2\left(1+\binom{n}{1}+\binom{n}{2}+\cdots+\binom{n}{s}\right)$$

Lower Bounds: a side note

The lower bounds assume arbitrary placement of bit errors.

In practice errors are likely to have patterns:
maybe evenly spaced, or clustered:



Can we do better if we assume **regular errors**?

We will come back to this later when we talk about **Reed-Solomon** codes. This is a big reason why Reed-Solomon codes are used much more than Hamming-codes.

Linear Codes

If $\sum$ is a field, then $\sum^n$ is a vector space

Definition: C is a linear code if it is a linear subspace of $\sum^n$ of dimension k.

This means that there is a set of k independent vectors $v_i \in \sum^n$ $(1 \leq i \leq k)$ that span the subspace.

i.e. every codeword can be written as:

$c = a_1 v_1 + a_2 v_2 + \ldots + a_k v_k$ where $a_i \in \sum$

"Linear": the sum of two codewords is a codeword.

Linear Codes

Vectors for the $(7,4,3)_2$ Hamming code:

		m_7	m_6	m_5	p_4	m_3	p_2	p_1
v_1	=	1	0	0	1	0	1	1
v_2	=	0	1	0	1	0	1	0
v_3	=	0	0	1	1	0	0	1
v_4	=	0	0	0	0	1	1	1

Another way to see that d = 3 for Hamming codes?

What is the least Hamming weight among non-zero codewords?

Distance of code = least weight codeword (for linear codes)

Generator and Parity Check Matrices

Generator Matrix:

A k x n matrix **G** such that: $C = \{ xG \mid x \in \sum^k \}$

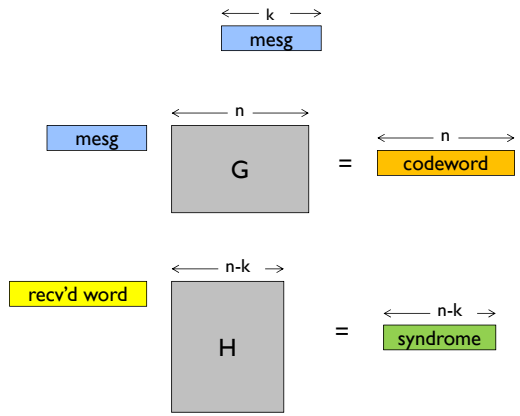
Made from stacking the spanning vectors

Parity Check Matrix:

An (n – k) x n matrix **H** such that: $C = \{y \in \sum^n \mid Hy^T = 0\}$

(Codewords are the null space of H.)

These **always exist for linear codes**



if syndrome = 0, received word = codeword

else have to use syndrome to get back codeword ("decode")

Advantages of Linear Codes

- Encoding is efficient (vector-matrix multiply)
- Error detection is efficient (vector-matrix multiply)
- **Syndrome** (Hy^T) has error information
- How to decode? In general, have q^{n-k} sized table for decoding (one for each syndrome). Useful if n-k is small, else want other approaches.

Example and "Standard Form"

For the Hamming (7,4,3) code:

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

By swapping columns 4 and 5 it is in the form I_k,A.

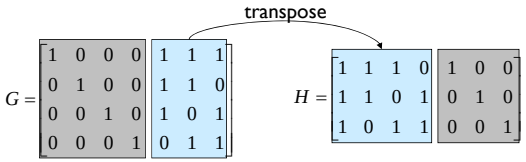
$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

A code with a matrix in this form is **systematic**, and G is in "**standard form**"

Relationship of G and H

Theorem: For binary codes, if G is in standard form $[I_k\ A]$ then $H = [A^T\ I_{n-k}]$

Example of (7,4,3) Hamming code:



Proof that H is a Parity Check Matrix (over $\mathbb{F}_2$)

Suppose that x is a message. Then

$H(xG)^T = H(G^Tx^T) = (HG^T)x^T = (A^TI_k+I_{n-k}A^T)x^T =$
$(A^T + A^T)x^T = 0$

Conversely, suppose that $Hy^T = 0$. Then for each $1 \le i \le n-k$
$A^T_{i,*} \bullet y^T_{[1..k]} + y^T_{k+i} = 0$
(where $A^T_{i,*}$ is row i of A^T and $y^T_{[1..k]}$ are the first k elements of y^T]).

Thus, $y_{[1..k]} \bullet A_{*,i} = y_{k+i}$ where $A_{*,i}$ is now column i of A, and $y_{[1..k]}$ are the first k elements of y, so $y_{[k+1\ldots n]} = y_{[1..k]}A$.

Consider $x = y_{[1..k]}$. Then $xG = [y_{[1..k]} \mid y_{[1..k]}A] = y$.

Hence if $Hy^T = 0$, y is the codeword for $x = y_{[1..k]}$.

Relationship of G and H

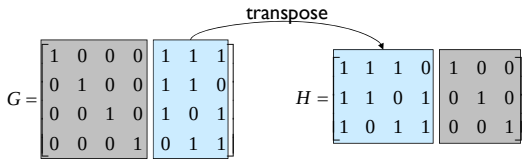
The above proof held only for $\mathbb{F}_2$.

For codes over a general field $\mathbb{F}_q$,
if G is of the standard form $[I_k, A]$
then the parity check matrix $H = [-A^T\ I_{n-k}]$

In the binary case, $-A = A$ and hence the principle is the same

The d of linear codes

Theorem: Linear codes have distance d if every set of (d-1) columns of **H** are linearly independent, but there is a set of d columns that are linearly dependent.



High level idea: for linear codes, distance equals least weight of non-zero codeword. And each codeword gives some collection of columns that must sum to zero.

The d of linear codes

Theorem: Linear codes have distance d if every set of (d-1) columns of **H** are linearly independent, but there is a set of d columns that are linearly dependent.

If some set S of d-1 columns were linearly dependent, then $\sum_{i \in S} c_k H_k = 0$

But then y which has zeroes on coordinates outside S, and c_i for each coordinate $i \in S$ satisfies $Hy = 0$, so is codeword of weight < d, a contradiction.

Conversely, distance d means there's a codeword y of weight d, which means $Hy = 0$ and hence the columns of H for the non-zero coordinates of y are linear dependent.

Dual Codes

For every code with
$G = [I_k \; A]$ and $H = [A^T \; I_{n-k}]$
we have a **dual code** with
$G = [I_{n-k} \; A^T]$ and $H = [A \; I_k]$


Jacques Hadamard (1865-1963)

The dual of the Hamming codes are the **binary "simplex" or Hadamard codes**: $(2^r\text{-}1, r, 2^{r-1})$

Dual Codes

For every code with
$G = [I_k \; A]$ and $H = [A^T \; I_{n-k}]$
we have a **dual code** with
$G = [I_{n-k} \; A^T]$ and $H = [A \; I_k]$


Irving Reed


David Muller

The dual of the Hamming codes are the **binary "simplex" or Hadamard codes**: $(2^r\text{-}1, r, 2^{r-1})$ codes

The dual of the extended Hamming codes are the **first-order Reed-Muller** codes.

Note that these codes are highly redundant, with very low rate. Where would these be useful?

NASA Mariner

Deep space probes from 1969-1977.

Mariner 10 shown

Used (32,6,16) Reed Muller code (r = 5)

Rate = 6/32 = .1875 (only 1 out of 5 bits are useful)

Can fix up to 7 bit errors per 32-bit word

Dual Codes

For every code with

$G = [I_k\ A]$ and $H = [A^T\ I_{n-k}]$

we have a **dual code** with

$G = [I_{n-k}\ A^T]$ and $H = [A\ I_k]$

Dual of $(2^r\text{-}1, 2^r\text{–}r\text{–}1, 3)$ Hamming code has generator matrix

$$G = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

N.b.: every non-zero r-bit vector appears as a column.

Lemma: this is a $(2^r - 1, r, 2^{r-1})$ code.

How to find the error locations

Hy^T is called the **syndrome** (no error if 0).

In **general** we can find the error location by creating a table that maps each syndrome to a set of error locations.

Theorem: assuming $s \leq (d\text{-}1)/2$ errors, every syndrome value corresponds to a unique set of error locations.

Proof: HW exercise.

Keep table of all these syndrome values. Has q^{n-k} entries, each of size at most n (i.e. keep a bit vector of locations).

Generic algorithm: not efficient!! (Better for special codes.)

Reed-Solomon Codes

Irving S. Reed and Gustave Solomon

PDF-417

QR code

Aztec code

DataMatrix code

All 2-dimensional Reed-Solomon bar codes

images: wikipedia

Reed-Solomon Codes in the Real World

(204,188,17)$_{256}$: ITU J.83(A)[2]
(128,122,7)$_{256}$: ITU J.83(B)
(255,223,33)$_{256}$: Common in Practice
- Note that they are all byte based (i.e., symbols are from GF(2^8)).

Decoding rate on 1.8GHz Pentium 4:
- (255,251) = 89Mbps
- (255,223) = 18Mbps

Dozens of companies sell hardware cores that operate 10x faster (or more)
- (204,188) = 320Mbps (Altera decoder)

Applications of Reed-Solomon Codes

- **Storage**: CDs, DVDs, "hard drives",
- **Wireless**: Cell phones, wireless links
- **Sateline and Space**: TV, Mars rover, Voyager,
- **Digital Television**: DVD, MPEG2 layover
- **High Speed Modems**: ADSL, DSL, ..

Good at handling burst errors.

Other codes are better for random errors.
- e.g., Gallager codes, Turbo codes