HAMMING AND GOLAY CODES

Satish Kumar Buddha

November 7, 2011

Error correcting codes

- In Information theory and coding theory, error detection and correction or error control are techniques that enable reliable delivery of digital data over unreliable communication channels.

Error correcting codes

- In Information theory and coding theory, error detection and correction or error control are techniques that enable reliable delivery of digital data over unreliable communication channels.
- Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow detecting such errors, while error correction enables reconstruction of the original data.

Definition

- Error detection is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver.

Definition

- Error detection is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver.
- Error correction is the detection of errors and reconstruction of the original, error-free data.

Error detection schemes

1. Repetition codes
2. Parity bits
3. Checksums
4. Cyclic redundancy checks (CRCs)
5. Cryptographic hash functions
6. Error-correcting codes

1. Automatic repeat request
2. Error-correcting code
3. Hybrid schemes

HISTORY

In the late $1940's$ Richard Hamming recognized that the further evolution of computers required greater reliability, in particular the ability to not only detect errors, but correct them. His search for error-correcting codes led to the Hamming Codes, perfect 1-error correcting codes, and the extended Hamming Codes, 1-error correcting and 2-error detecting codes.

USES

1. Hamming Codes are still widely used in computing, telecommunication, and other applications.
2. Hamming Codes also applied in:

- Data compression.
- Some solutions to the popular puzzle The Hat Game.
- Block Turbo Codes.

A $[7,4]$ binary Hamming Code.

- Let our codeword be $(x_1, x_2, \ldots x_7) \epsilon F_2^7$

A $[7, 4]$ binary Hamming Code.

- Let our codeword be $(x_1, x_2, \dots x_7) \epsilon F_2^7$
- (x_3, x_5, x_6, x_7) are chosen according to the message (perhaps the message itself is (x_3, x_5, x_6, x_7)).

A $[7,4]$ binary Hamming Code.

- Let our codeword be $(x_1, x_2, \ldots x_7) \epsilon F_2^7$
- (x_3, x_5, x_6, x_7) are chosen according to the message (perhaps the message itself is (x_3, x_5, x_6, x_7)).
- $x_4 := x_5 + x_6 + x_7 (mod\ 2)$

A $[7,4]$ binary Hamming Code.

- Let our codeword be $(x_1, x_2, \ldots x_7) \epsilon F_2^7$
- (x_3, x_5, x_6, x_7) are chosen according to the message (perhaps the message itself is (x_3, x_5, x_6, x_7)).
- $x_4 := x_5 + x_6 + x_7 (mod\ 2)$
- $x_2 := x_3 + x_6 + x_7$

A $[7, 4]$ binary Hamming Code.

- Let our codeword be $(x_1, x_2, \dots x_7) \epsilon F_2^7$
- (x_3, x_5, x_6, x_7) are chosen according to the message (perhaps the message itself is (x_3, x_5, x_6, x_7)).
- $x_4 := x_5 + x_6 + x_7 (mod\ 2)$
- $x_2 := x_3 + x_6 + x_7$
- $x_1 := x_3 + x_5 + x_7$

$[7,4]$ binary Hamming codewords

$$
\begin{array}{ccc}
(0\ 0\ 0\ 0) & \rightarrow & (0\ 0\ 0\ 0\ 0\ 0\ 0) \\
(0\ 0\ 0\ 1) & \rightarrow & (1\ 1\ 0\ 1\ 0\ 0\ 1) \\
(0\ 0\ 1\ 0) & \rightarrow & (0\ 1\ 0\ 1\ 0\ 1\ 0) \\
(0\ 0\ 1\ 1) & \rightarrow & (1\ 0\ 0\ 0\ 0\ 1\ 1) \\
(0\ 1\ 0\ 0) & \rightarrow & (1\ 0\ 0\ 1\ 1\ 0\ 0) \\
(0\ 1\ 0\ 1) & \rightarrow & (0\ 1\ 0\ 0\ 1\ 0\ 1) \\
(0\ 1\ 1\ 0) & \rightarrow & (1\ 1\ 0\ 0\ 1\ 1\ 0) \\
(0\ 1\ 1\ 1) & \rightarrow & (0\ 0\ 0\ 0\ 1\ 1\ 1) \\
 & \vdots &
\end{array}
$$

A $[7, 4]$ binary Hamming Code

- Let $a = x_4 + x_5 + x_6 + x_7 (= 1 \textit{ iff one of these bits is in error})$

A [7, 4] binary Hamming Code

- Let $a = x_4 + x_5 + x_6 + x_7 (= 1 \textit{ iff one of these bits is in error})$
- Let $b = x_2 + x_3 + x_6 + x_7$

A $[7, 4]$ binary Hamming Code

- Let $a = x_4 + x_5 + x_6 + x_7 (= 1 \textit{ iff one of these bits is in error})$
- Let $b = x_2 + x_3 + x_6 + x_7$
- Let $c = x_1 + x_3 + x_5 + x_7$

A $[7,4]$ binary Hamming Code

- Let $a = x_4 + x_5 + x_6 + x_7 (= 1 \textit{ iff one of these bits is in error})$
- Let $b = x_2 + x_3 + x_6 + x_7$
- Let $c = x_1 + x_3 + x_5 + x_7$
- If there is an error (assuming at most one) then *abc* will be binary representation of the subscript of the offending bit.

A $[7, 4]$ binary Hamming Code

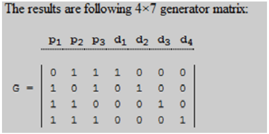
- If $(y_1, y_2 \dots y_7)$ is received and $abc \neq 000$, then we assume the bit abc is in error and switch it. If $abc = 000$, we assume there were no errors (so if there are three or more errors we may recover the wrong codeword).

Definition: Generator and Check Matrices

- For an $[n, k]$ linear code, the generator matrix is a $k\dot{n}$ matrix for which the row space is the given code.
- A check matrix for an $[n, k]$ is a generator matrix for the dual code. In other words, an $(n-k)\dot{k}$ matrix M for which $M\ x = 0$ for all x in the code.
- An (n, k, d) lexicode can encode up to k data bits in n bits total by including $n-k$ additional check bits. It can detect up to $d-1$ wrong bits and correct up to $(d-1)/2$ wrong bits. The values for a lexicode of dimension d all differ by at least d bits.

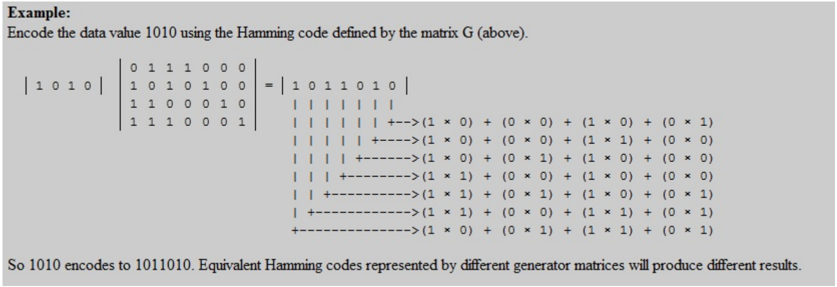
A Construction for binary Hamming Codes

1. Define d to be the 1×4 vector [d1 d2 d3 d4]
2. It's possible to create a 4×7 generator matrix [G] such that the product modulo 2 of d and [G] (d[G]) is the desired 1×7 Hamming code word. Here's how it's done:
3. Represent each data bit with a column vector as follows:
4. Represent each parity bit with a column vector containing a 1 in the row corresponding to each data bit included in the computation and a zero in all other rows.
5. Create a generator matrix, $[G]$, by arranging the column vectors from the previous steps into a 4×7 matrix such that the columns are ordered to match their corresponding bits in a code word.



The results are following 4×7 generator matrix:

$$G = \begin{array}{ccccccc} p_1 & p_2 & p_3 & d_1 & d_2 & d_3 & d_4 \\ \hline 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}$$



Example:
Encode the data value 1010 using the Hamming code defined by the matrix G (above).

```
         | 0 1 1 1 0 0 0 |
| 1 0 1 0 | | 1 0 1 0 1 0 0 | = | 1 0 1 1 0 1 0 |
         | 1 1 0 0 0 1 0 |    | | | | | | |
         | 1 1 1 0 0 0 1 |    | | | | | | +-->(1 × 0) + (0 × 0) + (1 × 0) + (0 × 1)
                              | | | | | +---->(1 × 0) + (0 × 0) + (1 × 1) + (0 × 0)
                              | | | | +------>(1 × 0) + (0 × 1) + (1 × 0) + (0 × 0)
                              | | | +-------->(1 × 1) + (0 × 0) + (1 × 0) + (0 × 0)
                              | | +---------->(1 × 1) + (0 × 1) + (1 × 0) + (0 × 1)
                              | +------------>(1 × 1) + (0 × 0) + (1 × 1) + (0 × 1)
                              +-------------->(1 × 0) + (0 × 1) + (1 × 1) + (0 × 1)
```

So 1010 encodes to 1011010. Equivalent Hamming codes represented by different generator matrices will produce different results.

- Hamming code can be written more compactly as follows. It is a linear code, that is, the transmitted codeword $'t'$ can be obtained from the source sequence $'s'$ by a linear operation, $t = transpose(G) * s$ where is the ‘generator matrix’ of the code.

$$\mathbf{G^T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix},$$

Syndrome Decoding

- Let y $= (y_1, y_2, \ldots y_n)$ be a received codeword.

Syndrome Decoding

- Let y $= (y_1, y_2, \ldots y_n)$ be a received codeword.
- The syndrome of y is $S := L_r yT$. If S=0 then there was no error. If S$\neq$ 0 then S is the binary representation of Some integers $1 \leq t \leq n = 2r - 1$ and the intended codeword is x $=$ $(y_1 \ldots y_r + 1 \ldots y_n)$.

Example Using L_3

- Suppose (1010010) is received.

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

100 is 4in binary, so the intended codeword was (1011010).

Extended $[8,4]$ binary Hamming Code.

- As with the $[7,4]$ binary Hamming Code:
- x_3, x_5, x_6, x_7 are chosen according to the message.
- $x_4 := x_5 + x_6 + x_7$
- $x_2 := x_3 + x_6 + x_7$
- $x_1 := x_3 + x_5 + x_7$

- $x_0 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$. i.e., the new bit makes the sum of all the bits zero. x_0 is called a parity check.

Extended $[8, 4]$ binary Hamming Code.

- As with the $[7, 4]$ binary Hamming Code:
- x_3, x_5, x_6, x_7 are chosen according to the message.
- $x_4 := x_5 + x_6 + x_7$
- $x_2 := x_3 + x_6 + x_7$
- $x_1 := x_3 + x_5 + x_7$
- Add a new bit x_0 such that
- $x_0 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$. i.e., the new bit makes the sum of all the bits zero. x_0 is called a parity check.

Extended binary Hamming Code

- The minimum distance between any two codewords is nowv 4, so an extended Hamming Code is a 1-error correcting and 2-error detecting code.
- The general construction of a $[2_r, 2_r - 1 - r]$ extended code from a $[2_r - 1, 2_r - 1 - r]$ binary Hamming Code is the same: add a parity check bit.

Perfect 1-error correcting

- Hamming Codes are perfect 1-error correcting codes. That is, any received word with at most one error will be decoded correctly and the code has the smallest possible size of any code that does this
- For a given r, any perfect 1-error correcting linear code of length $n = 2r - 1$ and dimension $n - r$ is a Hamming Code.

Applications

- Data compression.
- Turbo Codes.
- The Hat Game.

Data Compression

- Hamming Codes can be used for a form of lossy compression.
- If $n = 2r - 1$ for some r, then any n-tuple of bits x is within distance at most 1 from a Hamming codeword c. Let G be a generator matrix for the Hamming Code, and$mG = c$.
- For compression, store x as m. For decompression, decode m as c. This saves r bits of space but corrupts (at most) 1 bit.

The Hat Game

- A group of n players enter a room whereupon they each receive a hat. Each player can see everyone elses hat but not his own.
- The players must each simultaneously guess a hat color, or pass.
- The group loses if any player guesses the wrong hat color or if every player passes.
- Players are not necessarily anonymous, they can be numbered.

Golay code

- Marcel Golay was a successful mathematician and information theorist who was better known for his contribution to real world applications of mathematics.
- He generalized of perfect Binary Hamming Codes, inventor of the Golay Cell, and developing the Golay-Codes .
- Golays sought the perfect code. Perfect codes are considered the best codes and are of much interest to mathematicians.

Perfect codes

- A code C consisting of N code words of length N containing letters from an alphabet of length q, where the minimum distance $d = 2e + 1$ is said to be perfect if:

$$\sum_{i=0}^{e}\binom{n}{i}(q-1)^{i}=\frac{q^{n}}{N}$$

There are certain trivial perfect codes:

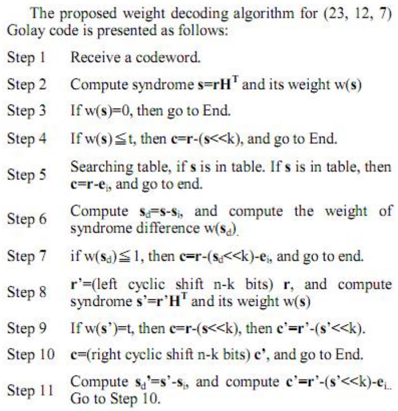
- Any code consisting of a single code word.
- Binary repetition codes of odd length. Any other perfect codes are considered non-trivial.

- In 1949 Marcel Golay he noticed that:

$$\binom{23}{0}+\binom{23}{1}+\binom{23}{2}+\binom{23}{3}=2^{11}$$

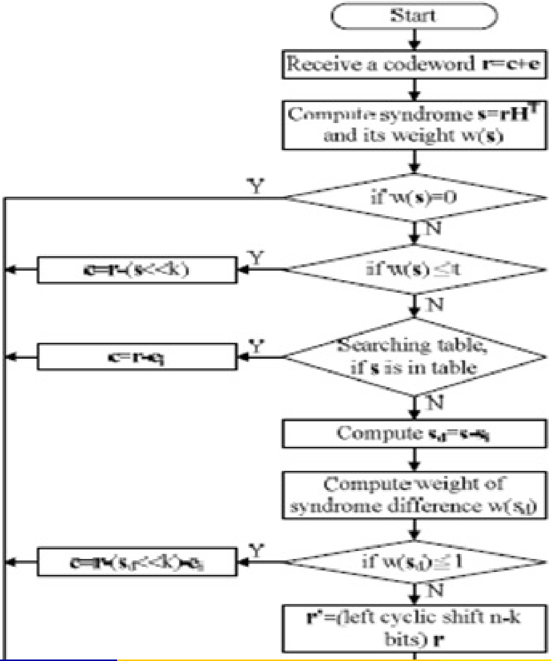
It indicated to him that the possibility of a (23, 12) perfect binary code existed that could correct 3 or fewer errors . This led to the binary form of the Golay code. It is one of the few examples of a non-trivial perfect code. This is the only known code capable of correcting any combination of three or fewer random errors in a block of 23 elements.

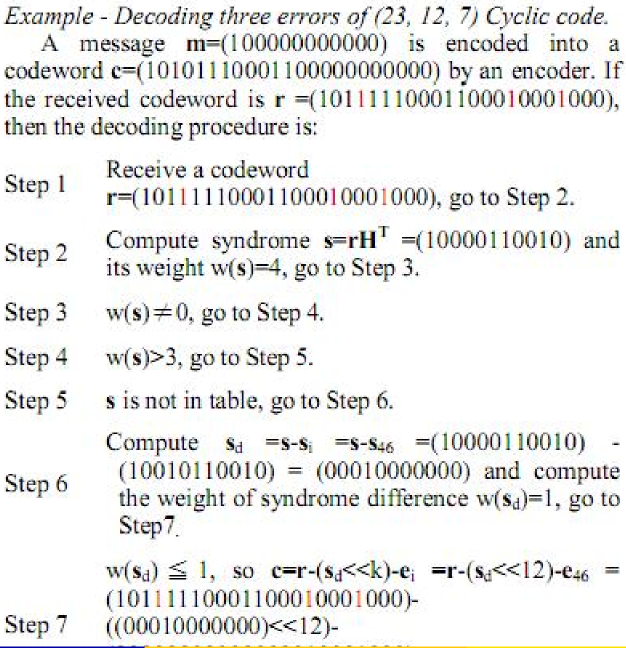
Decoding algorithm for golay code $(23, 12, 7)$



The proposed weight decoding algorithm for (23, 12, 7) Golay code is presented as follows:

Step 1	Receive a codeword.
Step 2	Compute syndrome $\mathbf{s}=\mathbf{r}\mathbf{H}^T$ and its weight w($\mathbf{s}$)
Step 3	If w($\mathbf{s}$)=0, then go to End.
Step 4	If w($\mathbf{s}$)$\leqq$t, then $\mathbf{c}$=$\mathbf{r}$-($\mathbf{s}$<<k), and go to End.
Step 5	Searching table, if $\mathbf{s}$ is in table. If $\mathbf{s}$ is in table, then $\mathbf{c}$=$\mathbf{r}$-$\mathbf{e}_i$, and go to end.
Step 6	Compute $\mathbf{s}_d$=$\mathbf{s}$-$\mathbf{s}_i$, and compute the weight of syndrome difference w($\mathbf{s}_d$).
Step 7	if w($\mathbf{s}_d$)$\leqq$1, then $\mathbf{c}$=$\mathbf{r}$-($\mathbf{s}_d$<<k)-$\mathbf{e}_i$, and go to end.
Step 8	$\mathbf{r}$'=(left cyclic shift n-k bits) $\mathbf{r}$, and compute syndrome $\mathbf{s}$'=$\mathbf{r}$'$\mathbf{H}^T$ and its weight w($\mathbf{s}$)
Step 9	If w($\mathbf{s}$')=t, then $\mathbf{c}$=$\mathbf{r}$-($\mathbf{s}$<<k), then $\mathbf{c}$'=$\mathbf{r}$'-($\mathbf{s}$'<<k).
Step 10	$\mathbf{c}$=(right cyclic shift n-k bits) $\mathbf{c}$', and go to End.
Step 11	Compute $\mathbf{s}_d$'=$\mathbf{s}$'-$\mathbf{s}_i$, and compute $\mathbf{c}$'=$\mathbf{r}$'-($\mathbf{s}$'<<k)-$\mathbf{e}_i$. Go to Step 10.


Start
Receive a codeword r=c+e
Compute syndrome s=rH^T and its weight w(s)
if w(s)=0
Y
N
if w(s) ≤ t
Y
c=r-(s<<k)
N
Searching table, if s is in table
Y
c=r+q
N
Compute s_d=s+s_i
Compute weight of syndrome difference w(s_d)
if w(s_d) ≤ 1
Y
c=r+(s_d<<k)+e_i
N
r'=(left cyclic shift n-k bits) r



Example - Decoding three errors of (23, 12, 7) Cyclic code.

A message **m**=(100000000000) is encoded into a codeword **c**=(10101110001100000000000) by an encoder. If the received codeword is **r** =(10111110001100010001000), then the decoding procedure is:

Step 1	Receive a codeword **r**=(10111110001100010001000), go to Step 2.
Step 2	Compute syndrome $\mathbf{s}=\mathbf{r}\mathbf{H}^T$ =(10000110010) and its weight w(**s**)=4, go to Step 3.
Step 3	w(**s**)$\neq$0, go to Step 4.
Step 4	w(**s**)>3, go to Step 5.
Step 5	**s** is not in table, go to Step 6.
Step 6	Compute $\mathbf{s}_d = \mathbf{s}-\mathbf{s}_i = \mathbf{s}-\mathbf{s}_{46}$ =(10000110010) - (10010110010) = (00010000000) and compute the weight of syndrome difference w($\mathbf{s}_d$)=1, go to Step7.
Step 7	w($\mathbf{s}_d$) $\leqq$ 1, so $\mathbf{c}=\mathbf{r}-(\mathbf{s}_d<<k)-\mathbf{e}_i = \mathbf{r}-(\mathbf{s}_d<<12)-\mathbf{e}_{46}$ = (10111110001100010001000)- ((00010000000)<<12)-

Conclusion

A powerful weight decoder is developed to correct up to three errors in the $(23, 12, 7)$ Golay code. This decoding algorithm is based on the fact that it makes use of the properties of cyclic codes, the weight of syndrome, and the reduced lookup table.